Document downloaded from:

http://hdl.handle.net/10251/123283

This paper must be cited as:

Giraldo-Velásquez, FD.; España Cubillo, S.; Giraldo, WJ.; Pastor López, O. (2018). Evaluating the quality of a set of modelling languages used in combination: A method and a tool. Information Systems. 77:48-70. https://doi.org/10.1016/j.is.2018.06.002

The final publication is available at

http://doi.org/10.1016/j.is.2018.06.002

Copyright Elsevier

Additional Information

Accepted Manuscript

Evaluating the quality of a set of modelling languages used in combination: A method and a tool

Fáber D. Giraldo, Sergio España, William J. Giraldo, Óscar Pastor

PII: S0306-4379(16)30207-1
DOI: 10.1016/j.is.2018.06.002
Reference: IS 1316

To appear in: *Information Systems*

Received date: 11 May 2016
Revised date: 27 April 2018
Accepted date: 1 June 2018

Please cite this article as: Fáber D. Giraldo, Sergio España, William J. Giraldo, Óscar Pastor, Evaluating the quality of a set of modelling languages used in combination: A method and a tool, *Information Systems* (2018), doi: 10.1016/j.is.2018.06.002

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Highlights

- This work describes the MMQEF method for evaluating quality issues in MDE contexts.
- This method consider sets of modelling languages used in combination.
- Quality results from a taxonomic analysis over modelling languages and elements
- The method underlies over an Information System reference architecture.
- MMQEF address quality issues from the model-driven perspective.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Evaluating the quality of a set of modelling languages used in combination: A method and a tool

Fáber D. Giraldo[a,c,*], Sergio España[b], William J. Giraldo[a], Óscar Pastor[c]

[a]*SINFOCI Research group, Engineering Faculty, University of Quindío, Colombia.*
[b]*Department of Information and Computing Sciences, Utrecht University, The Netherlands.*
[c]*PROS Research Centre, Universitat Politècnica de València, Spain.*

Abstract

Modelling languages have proved to be an effective tool to specify and analyze various perspectives of enterprises and information systems. In addition to modelling language designs, works on model quality and modelling language quality evaluation have contributed to the maturity of the model-driven engineering (MDE) field. Although consolidated knowledge on quality evaluation is still relevant to this scenario, in previous works, we have identified misalignments between the topics that academia is addressing and the needs of industry in applying MDE, thus identifying some remaining challenges. In this paper, we focus on the need for a method to evaluate the quality of a set of modelling languages used in combination within a MDE environment. This paper presents MMQEF (*Multiple Modelling language Quality Evaluation Framework*), describing its foundations, presenting its method components and discussing its trade-offs.

Keywords:
Quality, model-driven engineering, information systems, the MMQEF method, reference taxonomy, model analytics.

ACCEPTED MANUSCRIPT

1. Introduction

When working with complex information systems (IS), developers must elicit, specify, and manage requirements from multiple stakeholders. It is often the case that several perspectives on information systems are combined [1]. Conceptual models have proven to be a valuable tool to accomplish this. The model-driven engineering (MDE) paradigm promotes the notion that conceptual models are the main artefacts during IS engineering. The quality of conceptual models is a key factor for the success of MDE development projects. To some extent, the quality of models is influenced by the quality of their corresponding modelling languages [2].

Currently, the implementation of complex information systems uses conceptual models for taming the heterogeneity and multiplicity of views that are involved in projects of this type. *Concerns*, *models*, and *views* are essential components of an IS [3]. However, reports about the adoption of the MDE initiatives indicate that MDE itself possesses open questions that affect its applicability, especially in organizational contexts. Reference [4] previously reported these gaps and open challenges, based on evidence from industrial and academic experience.

The freedom that MDE promotes for managing IS concerns leads to the formulation of alternatives at modelling levels without a precise rationale beyond the mere justification of the interests or needs of the authors. The adoption of MDE approaches has guided the development of many model-driven compliance initiatives. Although it emphasizes the use of models as primary artefacts of a software construction process, it causes a conceptual divergence in the support of specific views and/or concerns belonging to an IS. This phenomenon is strengthened by the semantics offered by the modelling languages because it can be too formal (complex) or not formal enough (i.e., without a proper semantics).

A clear example of this type of conceptual divergence is found in the modelling act for software and IS architecture specifications, in which multiple modelling languages have been reported for managing concerns including UML, profiling, stereotypes, other modelling languages, and domain-specific languages (DSLs). Because there is no universal rule for modelling the concerns involved in an architecture project, any mechanism for communicating decisions and rationale is valid even if it is not formally specified.

Despite the development of metamodeling, reference architectural frameworks, and ontological frameworks, the inability to consistently model all related and inherent views in an IS using a single metamodel or a single notation has been recognized and widely reported. In [5], it is shown how a single metamodel can be feasible only if the granularity and abstraction level of the viewpoints are similar, which is impossible to guarantee in a typical MDE scenario since the viewpoints often have different abstractions and granularities.

MDE proposes modelling languages as the new abstraction units; hence, the introduction of a new language in MDE environments should be as easy as creating a new class in a Java

*Corresponding author. Phone: (+57) 67359300 ext 995, 1003
Email addresses: `fdgiraldo@uniquindio.edu.co`, `fdgiraldo@pros.upv.es` (Fáber D. Giraldo), `s.espana@uu.nl` (Sergio España), `wjgiraldo@uniquindio.edu.co.` (William J. Giraldo), `opastor@pros.upv.es.` (Óscar Pastor)

ACCEPTED MANUSCRIPT

project [6]. In MDE projects, one can often find several proposals of languages, models, notations, and tools that manage specific concerns belonging to multiple views of an IS. However, in practice, several of these proposals are not applied owing to problems detected in their integration with a previous set of IS models.

Because of the increasing number of modelling languages and notations, some authors have proposed methods for assessing the quality of modelling languages. The rationale behind these proposals is that models are a means to express conceptions about some phenomena, to reason about such conceptions, and to communicate them to others. Reference [4] previously identified some quality evaluation frameworks for modelling languages, but the characteristics of MDE projects require additional features in the method. For example, in MDE projects, it is common for several modeling languages to be used in combination to specify different perspectives of a system. In such cases, the languages might overlap in expressiveness, or they could overlook some relevant concerns. Moreover, in MDE technological frameworks, the quality of modelling languages goes beyond the representational aspects: the language should be designed in a way that facilitates model transformations.

Current modelling language evaluation methods fail to identify the situations described above. They also do not take into account the most relevant features of the MDE itself in their formulation. This is a consequence of the many divergent interpretations of MDE. The lack of consensus regarding MDE produces attempts to add new notations and languages framed in MDE without the support of a rationale. *"There are so many ways to adopt an MDE approach that it is impossible to establish general conclusions about MDE itself"* [7].

The identified guidelines and frameworks do not evaluate the quality of models according to how they support translation of models into other models (even models that belong to the same viewpoint of an IS) or experiences that come from the application of an IS architecture reference to specific MDE environments.

We previously conducted a systematic literature review that revealed the trends and the discrepancies in the area of model and modelling language quality evaluation [4]. More importantly, we found that the industrial practice of MDE has some characteristics that pose open challenges to current modelling language evaluation methods. An important challenge comes from the fact that many MDE technological environments and projects rely on the combined use of several modelling languages that were not necessarily meant to be used together. Another critical issue is that models used in MDE projects are typically subject to (semi)automatic transformations, requiring the modelling languages to be appropriate for this purpose.

This paper proposes a solution with the potential to address some of the open challenges. We present *MMQEF*, a method to evaluate the quality of a set of modelling languages used in combination within an MDE context. The core of the method is a classification procedure that uses a reference taxonomy of IS concepts. As a proof of concept, we selected the Zachman framework [8], but the method could be applied using other IS architectures. We also describe the tool support that we developed and report on a practical application of the method and on an empirical validation, which proves that the Zachman framework is a good candidate for a reference taxonomy.

The remainder of this paper is structured as follows: Section 2 describes the theoretical background of the method for modelling language quality evaluation. Section 3 describes the method, explaining its main components and offering guidelines for practitioners. Section 4 addresses the applicability of the proposed method and provides rationale for some method design decisions. Section 5 concludes the paper and outlines future work.

2. Background

2.1. Conceptual approaches for addressing quality issues in MDE

An approach to address the model-driven issues of quality is *ontological analysis* (i.e., the assessment of modelling language elements w.r.t the guidance provided by ontology frameworks for ISs). Although the question about whether *ontological guidance results in better models* is an open issue [9], ontologies for IS, which include the Bunge-Wand-Weber (BWW) [10, 11], and the Unified Foundational Ontology (UFO) [12, 13], are commonly used to evaluate modelling languages in accordance with ontological constructs, establishing their completeness through a mapping process between modelling elements and ontological constructs. Other reported usages or examples of ontological analysis are the integration of modelling languages and the incorporation of modelling constructs inside previous proposals of modelling languages.

The main support provided by the ontological analysis approach is inference-based reasoning, in which the role of an *analyst* (or *designer*) of languages can assess the consequences of the constructs for a language in a specific representation of the real world that is interesting for users of ISs. Often, quality is referred to as an ontology-based solution owing to the difficulty of distinguishing between the *problem space* scope and the *solution space* scope that are associated with the *model-based* and *ontology-based* approaches, respectively [14].

Despite the great potential for inferences that ontologies provide to support reasoning on quality, they could influence (or alter) the essential features of modelling languages. In these approaches, modelling constructs must fix the ontological constructs, assuming that they are valid from the perspective of a specific community (but there are a plethora of ontologies). According to [15], if a model (and thus a modelling language) is ontologically incomplete, the analyst/designer role will have to augment the model(s) to ensure that the final computerized information system adequately reflects that portion of the real world that is intended to be simulated.

Another conceptual tool reported in the analytics process in model-driven contexts is *taxonomic analysis*. This tool provides guidance for specific modelling tasks through the classification of artefacts or approaches. The decisions on modelling

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

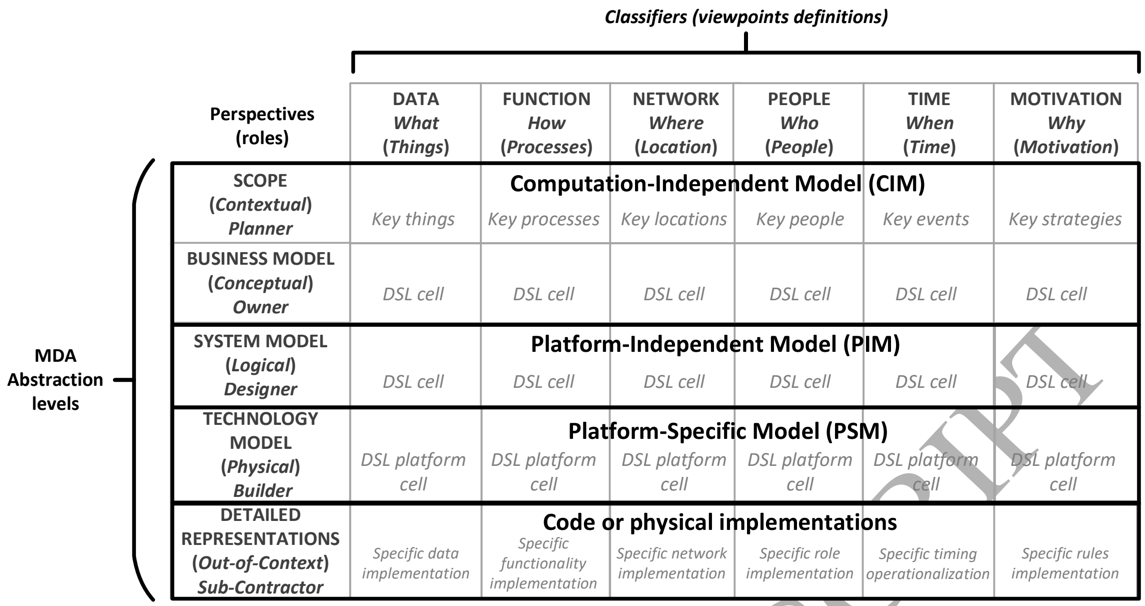

Classifiers (viewpoints definitions)
Perspectives (roles)
DATA What (Things)
FUNCTION How (Processes)
NETWORK Where (Location)
PEOPLE Who (People)
TIME When (Time)
MOTIVATION Why (Motivation)
MDA Abstraction levels
SCOPE (Contextual) Planner
Computation-Independent Model (CIM)
Key things
Key processes
Key locations
Key people
Key events
Key strategies
BUSINESS MODEL (Conceptual) Owner
DSL cell
SYSTEM MODEL (Logical) Designer
Platform-Independent Model (PIM)
DSL cell
TECHNOLOGY MODEL (Physical) Builder
Platform-Specific Model (PSM)
DSL platform cell
DETAILED REPRESENTATIONS (Out-of-Context) Sub-Contractor
Code or physical implementations
Specific data implementation
Specific functionality implementation
Specific network implementation
Specific role implementation
Specific timing operationalization
Specific rules implementation

Figure 1: Summary of the reference taxonomy.

artefacts[1] are made based on previous classification schemas. Examples of taxonomies that are applied in MDE contexts are: taxonomies for model transformations, taxonomies for MDE tools, taxonomies of model synchronization types, and taxonomies for managing the evolution of modelling languages.

Despite the high potential of taxonomic analysis for understanding the implications of model-driven practices, there are few reports of their use. In a way similar to ontological analysis, the process of taxonomic assessment can lead to subjective analysis due to its focus on specific features of the model-driven paradigm. Ontological and taxonomic analysis approaches must be used in a complementary way. Mechanisms for inference and classification are valuable strategies for managing challenges in the adoption of the model-driven paradigm because of their support for reasoning on models.

2.2. The reference taxonomy

The *Zachman framework* [8, 16] (hereinafter called the *reference taxonomy*) is a taxonomy derived from linguistic and philosophical sources, whose main purpose is to support a process for rationalizing the systemic use of IS elements to define an enterprise solution. The rationale process uses a procedure of classification. This taxonomy was conceived as an architecture proposal for the description of ISs that identifies the essential elements in a holistic system, which will be deployed in an organization. It established the basis for (and also influenced) current relevant standards such as ISO 42010 [3] (*software and systems architecture descriptions*) and frameworks for enterprise architecture.

[1]We use this term to refer to modelling languages and models generated from these languages.

Figure 1 summarizes the main features of the taxonomy reference. Basically, it is a classification language with a bidimensional configuration (taxonomy structure) that constitutes the *notation* of this classification language. Rows are the *abstraction levels* involved in an IS project; they are represented as roles that are relevant to levels of the organization and business (domain) down to the level of the specific technology for the implementation. Columns are philosophical *classifiers* [17][18] that are used to justify the elements involved in the construction of an IS regardless of the abstraction levels considered globally. The use of classifiers is commonly reported in the IS literature as an *analytic* conceptual tool for supporting reasoning and decisions. Classifiers are elements that derive implicit rules on views of IS (i.e., *viewpoints*).

Combinations of rows and columns (abstraction levels and classifiers, respectively) produce cells with specific purposes. They have a basic and implicit model that must be fulfilled for any modelling initiative that covers these purposes; this is the *DSL cell*. Figure 1 also depicts the alignment of the taxonomy with the MDA [19] architecture of reference according to a previous work reported in [20]. For the cells associated to the *Platform-specific Model* (the same as the *Technology model - physical* level of the taxonomy), a *DSL cell* is constrained by the implementation platform (e.g., models of Enterprise Java Beans for Java projects or models of Entity Framework for .Net platform); this is the *DSL platform cell*.

The combination of abstraction levels and classifiers gives taxonomic units (i.e., the cells in the grid) a unique meaning, scope, and intention, with associated *metaconcepts* that are specific for each cell. Some previous efforts have attempted to abstract the foundational concepts of the taxonomy using MOF-based metamodels and integration with IS foundational ontologies (see [20, 21, 22]); however, there is no consensus. Despite

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

this, these *metaconcepts* are implicit in the same specification of the taxonomy, so it is possible to infer the foundations of specific languages for each cell. In this way, the resulting classification is a comparison between the information of a modelling language (or information obtained from its instanced models) and the information contained in the metaconcepts of the cells.

The main feature of the reference taxonomy is its native support for the management of the semantics through its semantic bidimensional structure [23], in which conceptual models involved in an IS development process can be classified. The MDA guide 2.0 [24] focuses on the *semantic data* to perform model analytics procedures. However, it does not define any method to deduce these types of data in modelling artefacts. The semantics are defined w.r.t. a *semantic domain* and the mapping of the syntax within that domain. The semantic domain specifies the concepts that exist in the universe of discourse. This is a prerequisite for comparing semantic definitions [25].

The reference taxonomy per se does not define any relationship between elements of the framework. It defines only rules to classify information derived from conceptual models. Thus, any inference analysis is derived from the *classification* of the information about conceptual models that were used in a modelling procedure with the essential classifiers expected by the framework itself (*things, processes, locations, people, events, and strategies*). The reference taxonomy and ontological frameworks for ISs are complementary philosophical tools owing to the common presence of an IS architecture definition.

The most relevant contribution of the taxonomy for MDE contexts is the support for *reasoning* [26] on models and modelling languages in IS contexts (i.e., the underlying argument for the creation and use of related artefacts). Designers of modelling languages have a tool for decision-making to justify the purpose and intentions of the specific modelling efforts taking into account the current plethora of IS methods. This directly impacts the modelling language harmonization.

2.3. The support of the taxonomy for MDE quality issues

Given its formulation, the reference taxonomy works with conceptual models to represent IS phenomena that result in combining *abstractions* with *viewpoints*. The reference taxonomy manages the quality in IS projects with conceptual models through the accurate depiction of the relevant results from the classification act, the explicit treatment of the semantics, and the explicit consideration of transformations. The main goal of this conceptual tool is to relate conceptual things with representations on computers [16]. The taxonomic framework is a result of the abstractions; in this way, it facilitates any reasoning on the MDA architecture, which promotes software development by utilizing abstract hierarchical models.

The taxonomy itself does not propose any special procedure or methodology to evaluate models. However, because of this modelling support, quality issues of current models can be addressed by analytics procedures aligned to the classification itself. Examples of quality issues that can be addressed by the framework are the following:

Separation of concerns: the taxonomy promotes the selection of subsets in which decisions are applied. Instead of taming the complexity by using a global model of all of the cells, the taxonomy suggests that decisions must be made w.r.t. the scope of each cell.

Communication and suitability: the foundation of the taxonomy is the existence of a set of additives and complementary architectural representations (AR) for ISs. This is a criti-cal issue: in the *MDA foundation model document* [27], the metaclass *model* is associated explicitly with the term *architecture description* defined in the ISO 42010 standard (before the ISO 1471 standard). Thus, a reasoning mechanism is needed for discernment regarding the selection of a specific AR for an abstraction-viewpoint combination, given the variety of notations, methodologies, and languages. This discernment process involves the professional communication among the stakeholders that participate in a modelling effort for an IS project. The framework gives the freedom to use any approach or rationale for this analysis.

Model integration capabilities: the co-existence of several modelling initiatives must be evaluated to develop an IS in a consistent and optimized way. It applies the analysis to the capabilities offered by modelling languages to address goals related to the cells in the taxonomy. This analysis of all of the modelling languages involved in an IS development project improves modelling efforts, avoiding duplication of modelling approaches (i.e., using multiple models in different modelling languages to represent the same information), identifying complementary goals according to specific requirements of the modelling approaches, and identifying IS concerns that are not covered by the employed languages.

Model transformations: one of the most important challenges formulated by the taxonomy is to ensure that model transformations take place as a direct consequence of the addition of information in the interaction of abstractions and viewpoints. The *model mapping* feature (mentioned in the original MDA guide) occurs in the progressive changes in models that cross from higher to lower abstraction levels. Information from the computer-independent platform (CIM) level are enriched with constraints associated with lower levels. Thus, this leads to sufficient information to facilitate its implementation in a technical environment. In a similar way, the transformation of information in two different columns must be justified to support the criteria of the designer of the language to derive models from different essential properties or viewpoints (e.g., *time-location*, *data-process*). An explicit rationale about *why* and *how* information from a column can derive (or generate, or support) information for another column is critical in the features of modelling languages. The *traceability* evidence can be derived from analytics on the information classified by the reference taxonomy.

The taxonomic structure gives a useful and valuable set of information that is required to model any phenomena inside the scope of an IS [28]. The classification rules also guarantee consistency in any IS modelling activity.

ACCEPTED MANUSCRIPT

3. The MMQEF method

MMQEF (*Multiple Modelling language Quality Evaluation Framework*) is a method to evaluate the quality of modelling languages and models using a reference taxonomy for Information Systems (Section 2.2). Following the template for documentation of components for methods proposed in [29, 30], in this section we show the main considerations of the MMQEF method. We also address some of the most common questions regarding the use of the reference taxonomy in our method.

3.1. Purposes and preconditions

The main purpose of our method is to evaluate the quality of modelling languages used in MDE scenarios for IS development (i.e., the capability of any model artefact and the user language to represent an IS concern in the most appropiate way). Quality is the degree to which a model and/or modelling language has specific attributes to support essential features of IS and their implicit relations.

Our evaluation method supports semantic inferences and reasoning derived from the use of modelling languages in an IS construction process. The classification mechanism is used as a conceptual tool for determining the degree of support that a modelling language offers to represent a specific concern of the IS reference architecture (this is the reference taxonomy).

This method should be used in a model-driven project where:

- there is one or more modelling language(s) to support the concerns and viewpoints associated with the IS.
- the IS project integrally covers the MDA levels (CIM-PIM-PSM), and the transformations and mappings must be supported by a rationale.
- there is interest to know the real support of a modelling language when it will be applied to manage any IS concern.

The precondition for this method is to know the use of its bidimensional structure and its associated rules. The taxonomic structure of the reference taxonomy is compliant with the MDA levels [20]. This previous knowledge is important to apply and support classification procedures and decisions. The levels of the MDA specification (called abstraction levels) must be supported by specific modelling proposals in accordance with their nature and intention. In this way, a classification task could fail if the subjective criteria of the analyst are not aligned with the specific constraints of each taxonomy level.

3.2. Method components

The main component of the method is the *reference taxonomy*, which has a structure, a set of rules, and operations over the elements under classification. The structure offers an approach to explicitly manage abstraction levels jointly with classifiers that derive viewpoints. Rules of the taxonomic structure define the semantics associated with the act of modelling itself, contrasting the elements for classification with essential features that are associated with the IS.

The taxonomy reference defines seven rules. These rules are applied as originally formulated in [16] as follows:

R1 The columns do not have any specific order.
R2 Each column has a basic (*simple*) model. This implies that there is an *essential concept* for each column that answers the *question* of its associated column. The basic model constitutes the generic metamodel for any column.
R3 The basic model of each column is unique.
R4 Each row represents a perspective that is unique and different .
R5 Each cell is unique.
R6 The integration of the models of the cells in a row constitutes a *complete model* of this row.
R7 The logic of the framework is recursive (i.e., the essential models can explain themselves), and each cell could be analyzed with the use of the entire taxonomy.

The classification derived from this taxonomy requires the explicit consideration of the *technical issues implementation* of model artefacts under consideration. It is left to the discretion of the analyst/designer role to decide whether or not to consider the technical aspects. However, there is an explicit feature of the computational implementation that should sometimes be considered to demonstrate the feasibility of the modelling effort from a computational perspective.

We take the taxonomy and apply its specific procedures to contrast specific mechanisms to manage information that is offered by the modelling languages against the essential elements expected in the taxonomy by default.

The other required component is the information extracted from the modelling languages themselves or information from models. This required information pertains to the real use of models/modelling languages (expressed in the elements with which the user of languages interacts directly - e.g., concrete syntax and notations) and the semantics offered by the artefact under analysis. For this case, this type of semantic information is contained in MOF-based metamodels.

3.3. Cooperation principles

3.3.1. Structure

The evaluation method requires an organizational structure in which there is a global vision about the IS scope and goals. These structures have levels of architecture descriptions, crossing enterprise architecture, business architecture, and software architecture. These architectures are required by their implicit use, the presence of conceptual models, and the management of views/viewpoints that derive languages and methodologies to cover IS concerns.

3.3.2. Roles

Two roles are required for this evaluation method. The *modelling language analyst/designer* is responsible for proposing and/or conceiving some modelling language(s) to manage specific concerns in an IS project. Generally, this role could be assumed by architects, domain experts, or method engineers with

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

a global vision level of the overall project. A *modelling language final user* is identified also; this role refers to the stakeholders or domain experts who will use the language based on their intentions and interests in the IS.

3.4. The main quality evaluation method components

To represent the main components and procedures for our method, Figure 2 summarizes the method showing the inputs, outputs, and roles involved. The main inputs of the MMQEF method are the reference taxonomy, artefacts describing and clarifying the modelling language (i.e., language specifications such as metamodels, modelling guidelines, or even actual models that serve as examples), and the previous knowledge of the IS domain and the language itself. The output is the classification of modelling languages according to the organization of their information over the taxonomic structure. Inferred reasoning results from the classification itself.

The components of the method use the prior information about the domain (or knowledge of modelling tasks), and the information derived from the preconceptions from the participants in the modelling act [31]. Those inputs are contrasted considering the information expected for the cells of the taxonomy. This means that the classification is made by the contrast of information, and the reference taxonomy defines the principles that underlie the classification from an IS perspective.

Associated procedures of the method that are more specific are depicted using the *process-delivery diagram* (PDD) approach formulated in [32, 33]. Expanding the method depicted in Figure 2, we find an *Activity Diagram* with five main blocks associated with the evaluation procedure (Figure 3).

The activity diagram depicted in Figure 3 is for readability purposes. Each block represents one specific evaluation procedure supported by the reference taxonomy and the specific intention of the evaluation. A first set of activities is related to the *organization* that the language must have regarding the reference taxonomy, which in turn acts as a reference architecture for IS domains. Thus, this block checks for the architectonical structure as an essential property provided by the modelling language to integrally manage the concerns of an IS.

The second block of activities checks for the support provided for managing the incremental evolution of the information under modelling. In this way, the classification of model elements and derived information generate a *navigation model* over the bidimensional structure of the taxonomy, where the traces (or information) are identified that support the progressive evolution of the modelled information.

The third block of activities in Figure 3 verifies the capacities for the specification of *transformations* that the language should possess. It ensures that transformations can be conceived and occur from a semantic reasoning using the essential IS concerns of the classification, independent of constraints and rules imposed by languages and frameworks for model transformations.

A set of activities is considered for checking the *integration capacities* provided by the modelling language. The taxonomy serves to identify the IS concerns covered/not covered by the modelling language. For the uncovered concerns, activities verify whether the modelling language provides any mechanism for modelling these concerns by using one or more additional languages, and thus undertakes a full modelling effort that considers all relevant concerns for a specific IS project.

Finally, based on the individual classifications obtained for the modelling languages under evaluation, the *suitability* of modelling languages is analyzed if they share taxonomic concerns (cells) in their evaluation over the taxonomy. Individual results of the classification support the decisions about the most appropriate language for modelling a set of IS concerns.

With the activities related above, MMQEF gives a methodological orientation for deducing the quality evidence of modelling languages (from their properties) and modelling elements (from their use in an IS context); this evidence is derived from a classification procedure, which is the initial step of the method. MMQEF also provides a technical environment that enables inference of semantic deductions over the languages and elements under taxonomic analysis. Appendix A more thoroughly describes the activities and concepts associated with the MMQEF method following the PDD specification.

3.5. Derived MDE quality analytics

Some quality analytics procedures over modelling languages and model elements can be infered from the considerations presented above. These start with the classification of modelling elements and artefacts, either in the form of information from modelling language features or modelling language instances (representations). The classification pertains to the goal of the modelling artefacts with the scope of each abstraction/viewpoint combination.

3.5.1. Inference analysis derived from the Zachman taxonomy

Ontologies are supported by taxonomic structures [34]. Thus, inference reasoning is a result of a previous classification activity. The reference taxonomy itself is not an ontology [35], but this framework has the taxonomic structure required to promote inference reasoning for the use of models and modelling languages in an IS development process.

In analytic procedures on model artefacts, data semantics come from two main sources: the core concepts of languages (metamodel) and the information derived from model representations. The first relates core language concepts with the essential modelling dimensions expected in each abstraction/column combination. This action allows deduction of the IS concerns for which the modelling language was formulated. This is an analytics procedure from the language engineer[2].

Moreover, the second is a classification from a language user perspective, in which the *representations* become the main artefact to manage the interaction between the stakeholder and the IS. The classifications of the representations pertain to the purpose or goal perceived by the language users and the information that it contains. Inferences about it help the language engineer identify quality issues such as suitability, i.e., the harmonization of modelling initiatives for the design of languages without waiting until its implementation to find other modelling

[2]According to the roles defined in [36]

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

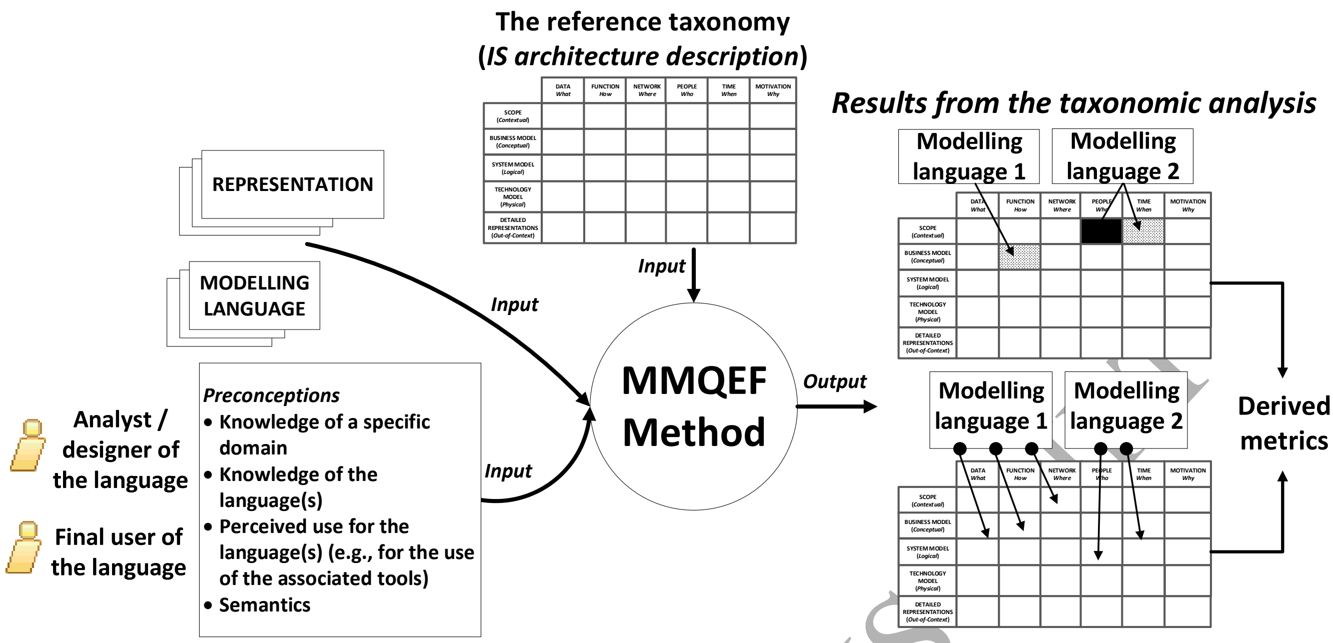

The reference taxonomy
(IS architecture description)
REPRESENTATION
MODELLING LANGUAGE
Analyst / designer of the language
Final user of the language
Preconceptions
• Knowledge of a specific domain
• Knowledge of the language(s)
• Perceived use for the language(s) (e.g., for the use of the associated tools)
• Semantics
Input
Input
Input
MMQEF Method
Output
Results from the taxonomic analysis
Modelling language 1
Modelling language 2
Derived metrics

Figure 2: Overview of the MMQEF method.

initiatives that address the same IS concern(s). This analysis implies that all stakeholders involved in the development of an IS must be identified before the formulation of models.

Any analytic procedure with representations requires information beyond the particular interpretation of the users of languages in accordance with their particular interpretation of the modelling elements. Thus, the *diagram* transcends from the *big picture* resource to an *information unit* with data about the expected/real intentional use. This is similar to the *pragmatic* of diagrams [37] under the model-driven way, in which it is important to obtain additional information that helps in making reasoning inferences.

3.5.2. Derived support for MDE quality evaluation

Support for assessing the quality of models and modelling languages is derived from the classification of the reference taxonomy. This is particularly relevant for model-driven practitioners because the evaluation is made directly on the modelling act itself rather than some adapted software quality practice [38] as is commonly reported in the MDE literature. The evaluation support is as follows:

A rationale about the organizational impact of the model-driven initiative(s): This refers to the degree of complete support that a model-driven initiative offers for an IS concern framed into an organizational context. Its analysis involves procedures related to the checking of the alignment of model-driven initiative purposes with the organizational goals, the degree of effective support of model-driven tools at computational-independent levels, the degree of understanding and use of models by organizational users, etc.

The native IS architecture on organizational constraints powered by the taxonomy facilitates the reasoning about the applicability of a model-driven approach in any organizational environment that uses IS. However, this analysis is not generic, and the specific business features in which models will be applied (e.g., business capabilities) must be considered to guarantee the compliance between the business concerns and model-driven approaches.

Analysis and design of transformations: The taxonomic analysis provides support for identifying the evolution of the information and the semantic relations generated. Thus, analysts of the language can propose decisions and considerations about model transformations before their implementation on a model transformation language.

In the case of model mappings (evolution from the PIM to the PSM level within a specific viewpoint), the effort will focus on the preservation and evolution of the high-level concepts until their preliminary technical implementation in accordance with the generic model of each viewpoint. Thus, the relations between the conceptual information and constraints imposed by the lower-level abstractions and perspectives are evident.

For transformation between models of different columns and the same row, there must be a semantic argument that supports the generation or derivation of models in different viewpoints and guarantees the complete depiction of the IS reality [16] from the row under analysis.

The taxonomic framework recognizes the *dependency* relation between cells in a given row, although these have relevant independence in their definition. These are relevant for the design of model transformations, especially when they are considered model co-evolution features in model management platforms. Dependency relationships generate conceptual mod-

ACCEPTED MANUSCRIPT

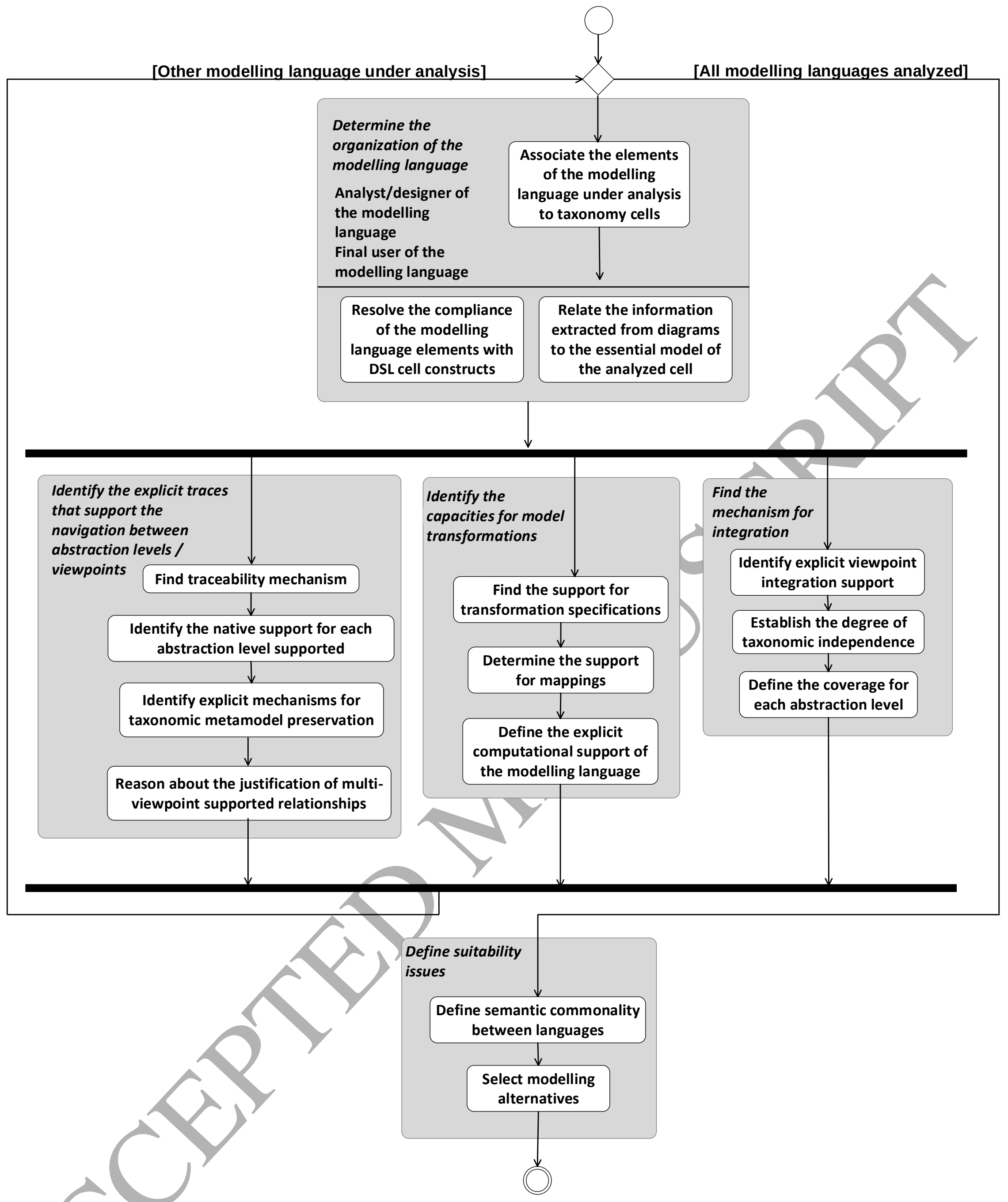

[Other modelling language under analysis]
[All modelling languages analyzed]
Determine the organization of the modelling language
Analyst/designer of the modelling language
Final user of the modelling language
Associate the elements of the modelling language under analysis to taxonomy cells
Resolve the compliance of the modelling language elements with DSL cell constructs
Relate the information extracted from diagrams to the essential model of the analyzed cell
Identify the explicit traces that support the navigation between abstraction levels / viewpoints
Find traceability mechanism
Identify the native support for each abstraction level supported
Identify explicit mechanisms for taxonomic metamodel preservation
Reason about the justification of multi-viewpoint supported relationships
Identify the capacities for model transformations
Find the support for transformation specifications
Determine the support for mappings
Define the explicit computational support of the modelling language
Find the mechanism for integration
Identify explicit viewpoint integration support
Establish the degree of taxonomic independence
Define the coverage for each abstraction level
Define suitability issues
Define semantic commonality between languages
Select modelling alternatives

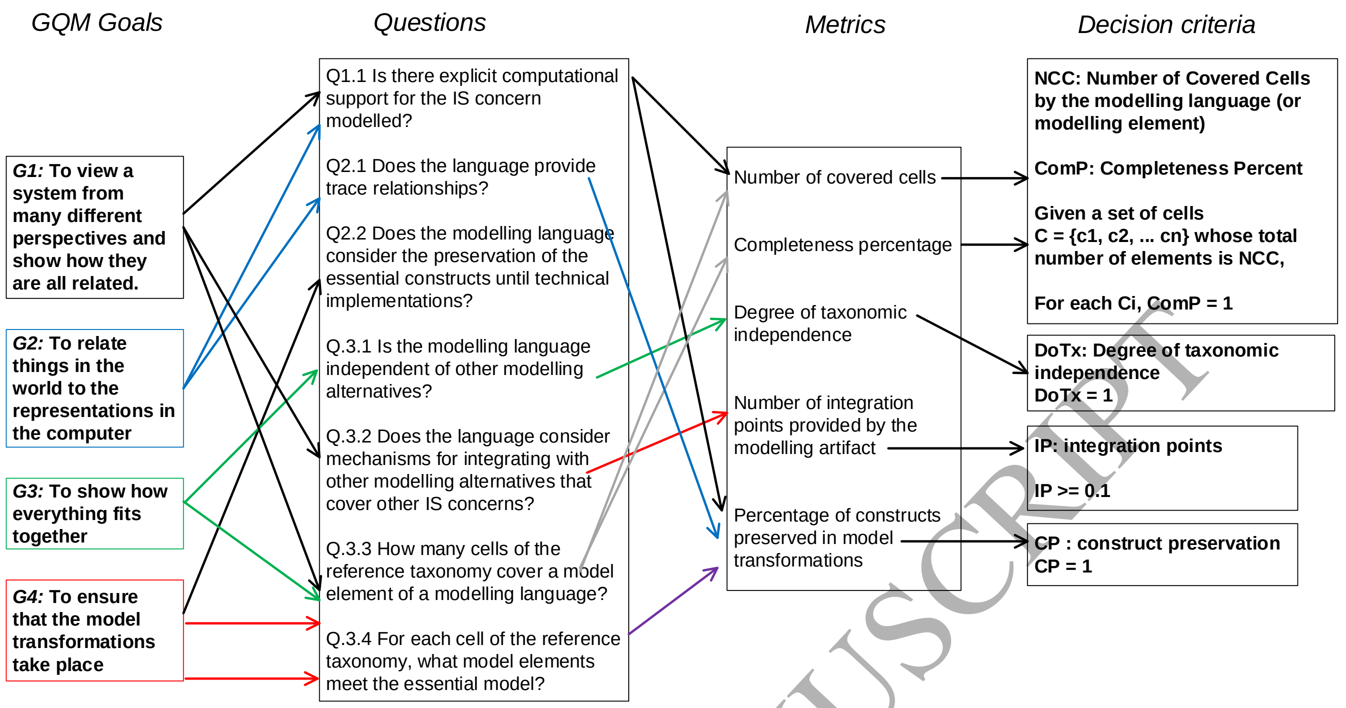
Figure 3: The main blocks of activities of the MMQEF method in the PDD convention.

els of transformations, posteriorly implemented on a modelling language transformation.

In scenarios of transformation, their underlying rationale is the *traceability information* that semantically manages the evolution of models. This information must be explicitly available to design and implement model transformations. This feature enables formulation of a process for consideration from the conception until the implementation and deployment of model transformations (in an analogous way to a software development process), without an explicit dependence only on the model transformation languages.

MDE metrics: Following the goal-question-metric approach [39](Figure 4), some metrics are formulated for supporting the quality evaluation of modelling languages with the reference taxonomy. For this, the goals were extracted directly from one of the original specifications of the taxonomic framework [16]. These relate to the development and deployment of models

ACCEPTED MANUSCRIPT



GQM Goals
Questions
Metrics
Decision criteria
G1: To view a system from many different perspectives and show how they are all related.
G2: To relate things in the world to the representations in the computer
G3: To show how everything fits together
G4: To ensure that the model transformations take place
Q1.1 Is there explicit computational support for the IS concern modelled?
Q2.1 Does the language provide trace relationships?
Q2.2 Does the modelling language consider the preservation of the essential constructs until technical implementations?
Q.3.1 Is the modelling language independent of other modelling alternatives?
Q.3.2 Does the language consider mechanisms for integrating with other modelling alternatives that cover other IS concerns?
Q.3.3 How many cells of the reference taxonomy cover a model element of a modelling language?
Q.3.4 For each cell of the reference taxonomy, what model elements meet the essential model?
Number of covered cells
Completeness percentage
Degree of taxonomic independence
Number of integration points provided by the modelling artifact
Percentage of constructs preserved in model transformations
NCC: Number of Covered Cells by the modelling language (or modelling element)
ComP: Completeness Percent
Given a set of cells C = {c1, c2, ... cn} whose total number of elements is NCC,
For each Ci, ComP = 1
DoTx: Degree of taxonomic independence
DoTx = 1
IP: integration points
IP >= 0.1
CP : construct preservation
CP = 1

Figure 4: Metrics for analytic reasoning derived using the GQM approach

over IS concerns, so these contribute to improving the applicability of a model-driven initiative. Questions define the scope of a modelling proposal, identifying other modelling proposals with similar purposes and anticipating consequences of the model initiative application. The derived metrics are as follows:

Number of covered cells (NCC): This refers to the total cells covered by the model element/modelling language. If number of covered cells belong to the same column, the modelling artefact has an explicit trace intention. However, if cells are from two or more columns, the modelling artefact must provide a semantic mechanism to derive relations between the viewpoints under consideration according to the goals of the modelling act.

Completeness percentage (ComP): For each cell covered by a model artefact, the purpose of the artefact is compared with the scope of the cell to determine its degree of compliance. This is useful to find modelling proposals in which the intention is not originally aligned with the scope, but the modeller/modelling language designer decides to use it for any reason. This also reveals *profiling trends* for modelling elements. This metric is as follows: $\forall\ cell \in NCC, ComP = 1$.

Degree of taxonomic independence (DoTx): This refers to the degree to which a classification obtained for a modelling language (or its associated modelling elements) is *distinctive*, which means that it is unique and clearly differs from classifications for other languages. The covered cells for the modelling language or element allow the main characteristics and scope of the classification to be deduced. If two or more languages are detected in an IS project that share common cells, the independence of the language for managing a specific concern is questioned, so a suitability reason is required to choose the most appropriate alternative for modelling these concerns. This decision could use other metrics, such as the *Completeness percentage - ComP*.

Number of integration points provided by the modelling artefact (IP): An analysis on the integration capabilities of modelling artefacts must be performed to find the semantic relations that support the coexistence of modelling efforts. This feature contributes to the design of model transformations for purposes of managing multiple instances of the IS model.

There is no universal prescription for the coverage that a modelling language should possess for the different abstractions/viewpoints of an IS. However, if a modelling language defines integration points, this feature is an explicit recognition for previous modelling initiatives that could better model some IS concerns, so the modelling language under analysis can focus on a specific IS concern and specialize its specific modelling approach.

Another direct consequence of this analysis is *suitability decisions*. This contrasts the degree of completeness of each modelling initiative w.r.t. the covered scope. This type of consensus could optimize the application of modelling proposals. From this, the suitability percentage of specific modelling proposals is derived.

Percentage of constructs preservation in model transformations (CP): This relates to the degree of preservation of essential

ACCEPTED MANUSCRIPT

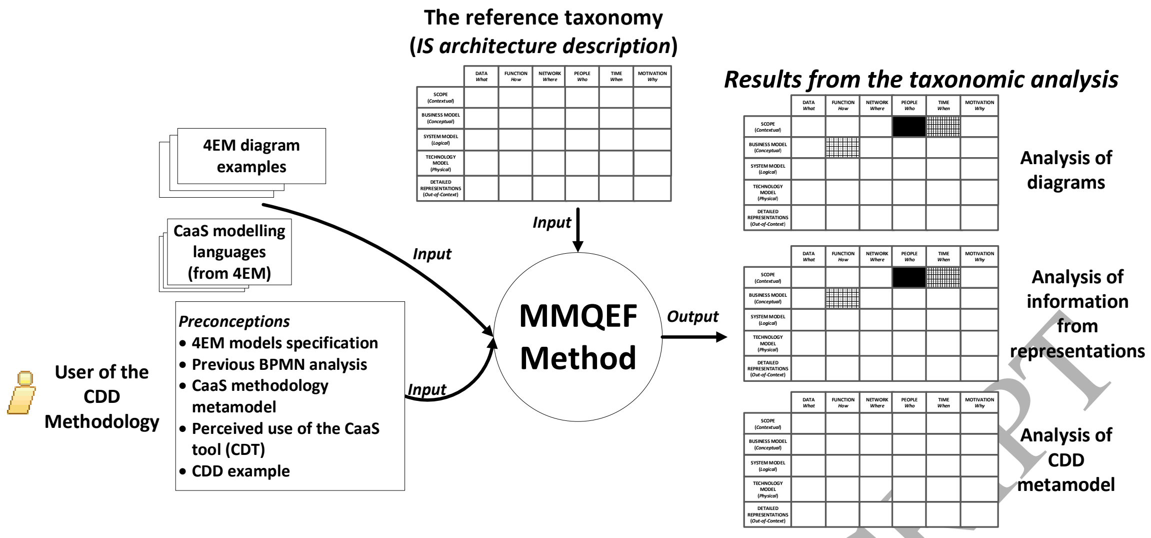
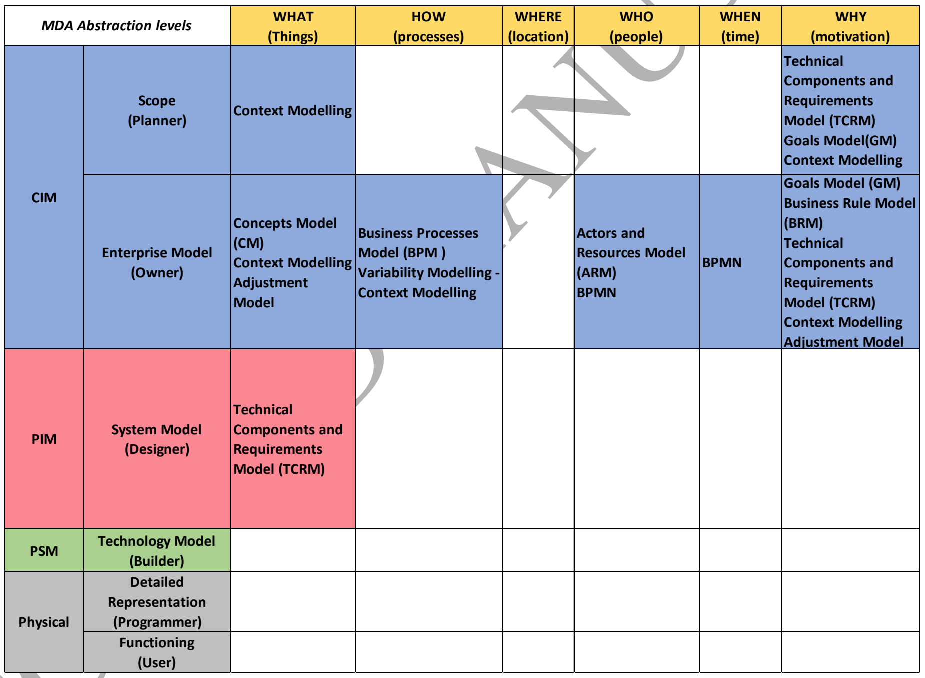

The reference taxonomy
(IS architecture description)
Results from the taxonomic analysis
4EM diagram examples
CaaS modelling languages (from 4EM)
Preconceptions
• 4EM models specification
• Previous BPMN analysis
• CaaS methodology metamodel
• Perceived use of the CaaS tool (CDT)
• CDD example
User of the CDD Methodology
Input
Input
Input
MMQEF Method
Output
Analysis of diagrams
Analysis of information from representations
Analysis of CDD metamodel

Figure 5: Summary of the application of the MMQEF method to the CDD methodology of the CaaS project.



MDA Abstraction levels		WHAT (Things)	HOW (processes)	WHERE (location)	WHO (people)	WHEN (time)	WHY (motivation)
CIM	Scope (Planner)	Context Modelling					Technical Components and Requirements Model (TCRM) Goals Model(GM) Context Modelling
	Enterprise Model (Owner)	Concepts Model (CM) Context Modelling Adjustment Model	Business Processes Model (BPM) Variability Modelling - Context Modelling		Actors and Resources Model (ARM) BPMN	BPMN	Goals Model (GM) Business Rule Model (BRM) Technical Components and Requirements Model (TCRM) Context Modelling Adjustment Model
PIM	System Model (Designer)	Technical Components and Requirements Model (TCRM)					
PSM	Technology Model (Builder)						
Physical	Detailed Representation (Programmer)						
	Functioning (User)						

Figure 6: Taxonomic analysis of the diagrams of the CDD methodology.

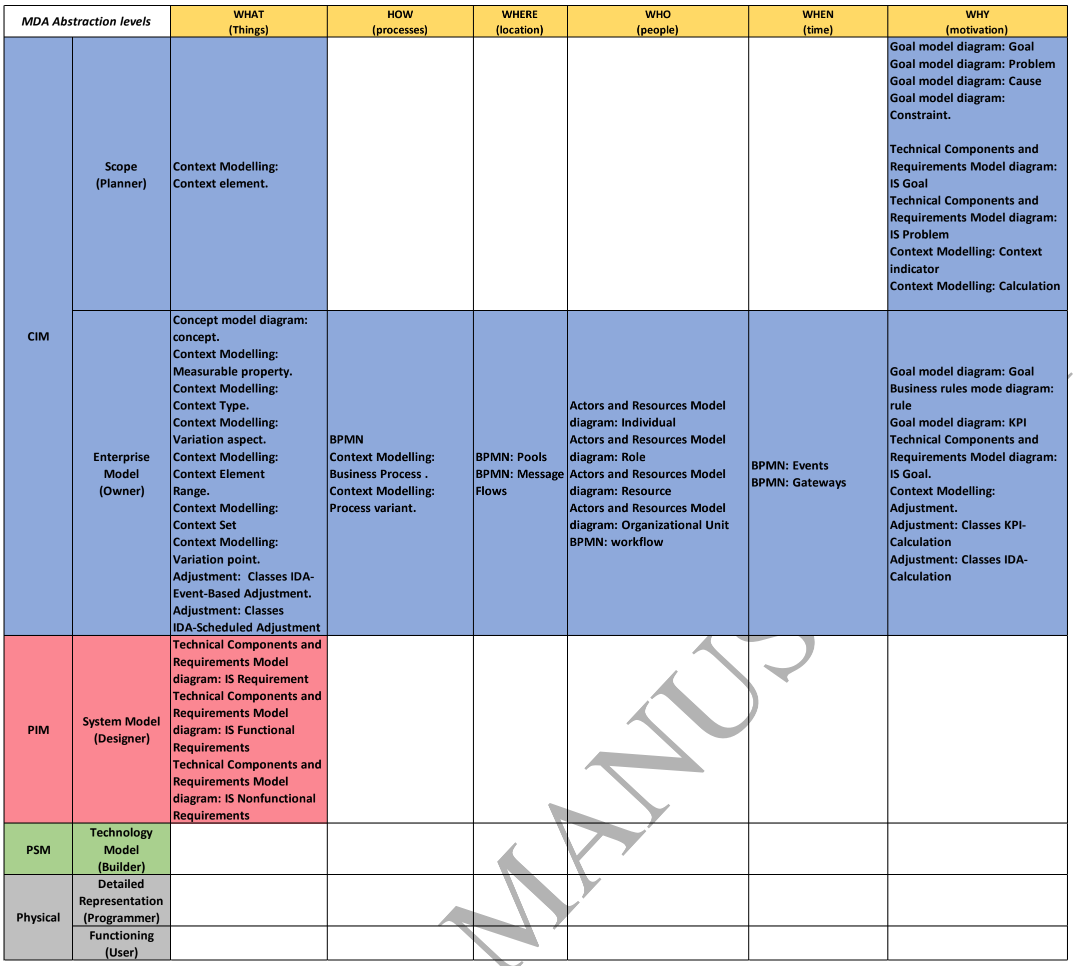
concepts during a high-low mapping until their technical implementation. The main goal of this metric is to avoid the loss of information at the crossroads between different abstraction levels / viewpoints so that the business concepts evolve until full technical implementation.

3.6. An example application of the MMQEF method

To demonstrate the applicability of our quality evaluation method, we use it to analyze the modelling languages that are used in a specific capability-driven development (CDD) scenario of the European project named *Capability as a Service in digital enterprises* (CaaS)[40]. The CaaS project proposes the CDD approach for digital enterprises to exploit the notion of *capability* as a means of design both for services and with services. Thus, CaaS elaborates an integrated approach consisting of methods, tools, and reusable best practices that allow companies to adjust their business services and information technology systems according to the changes in business context and technologies [41, 42].

The CDD methodology of the CaaS project uses modelling

ACCEPTED MANUSCRIPT



MDA Abstraction levels		WHAT (Things)	HOW (processes)	WHERE (location)	WHO (people)	WHEN (time)	WHY (motivation)
CIM	Scope (Planner)	Context Modelling: Context element.					Goal model diagram: Goal Goal model diagram: Problem Goal model diagram: Cause Goal model diagram: Constraint. Technical Components and Requirements Model diagram: IS Goal Technical Components and Requirements Model diagram: IS Problem Context Modelling: Context indicator Context Modelling: Calculation
	Enterprise Model (Owner)	Concept model diagram: concept. Context Modelling: Measurable property. Context Modelling: Context Type. Context Modelling: Variation aspect. Context Modelling: Context Element Range. Context Modelling: Context Set Context Modelling: Variation point. Adjustment: Classes IDA- Event-Based Adjustment. Adjustment: Classes IDA-Scheduled Adjustment	BPMN Context Modelling: Business Process . Context Modelling: Process variant.	BPMN: Pools BPMN: Message Flows	Actors and Resources Model diagram: Individual Actors and Resources Model diagram: Role Actors and Resources Model diagram: Resource Actors and Resources Model diagram: Organizational Unit BPMN: workflow	BPMN: Events BPMN: Gateways	Goal model diagram: Goal Business rules mode diagram: rule Goal model diagram: KPI Technical Components and Requirements Model diagram: IS Goal. Context Modelling: Adjustment. Adjustment: Classes KPI- Calculation Adjustment: Classes IDA- Calculation
PIM	System Model (Designer)	Technical Components and Requirements Model diagram: IS Requirement Technical Components and Requirements Model diagram: IS Functional Requirements Technical Components and Requirements Model diagram: IS Nonfunctional Requirements					
PSM	Technology Model (Builder)						
Physical	Detailed Representation (Programmer)						
	Functioning (User)						

Figure 7: Taxonomic analysis of the information extracted from diagrams of the CDD methodology.

languages to represent enterprise designs, context models, and patterns. These are based on the EKD [43] and 4EM [44] approaches to Enterprise Modelling. Thus, CaaS uses *goal models*, *process models*[3], *business rule models*, *concept models*, *actor/resource models*, and *technical component and requirement models*. The methodology also considers models for handling variability concerns. Those are models at the enterprise level, mitigating current limitations of specific platforms by the application of the CaaS approach. References [45, 46] report an example of an application over SOA platforms.

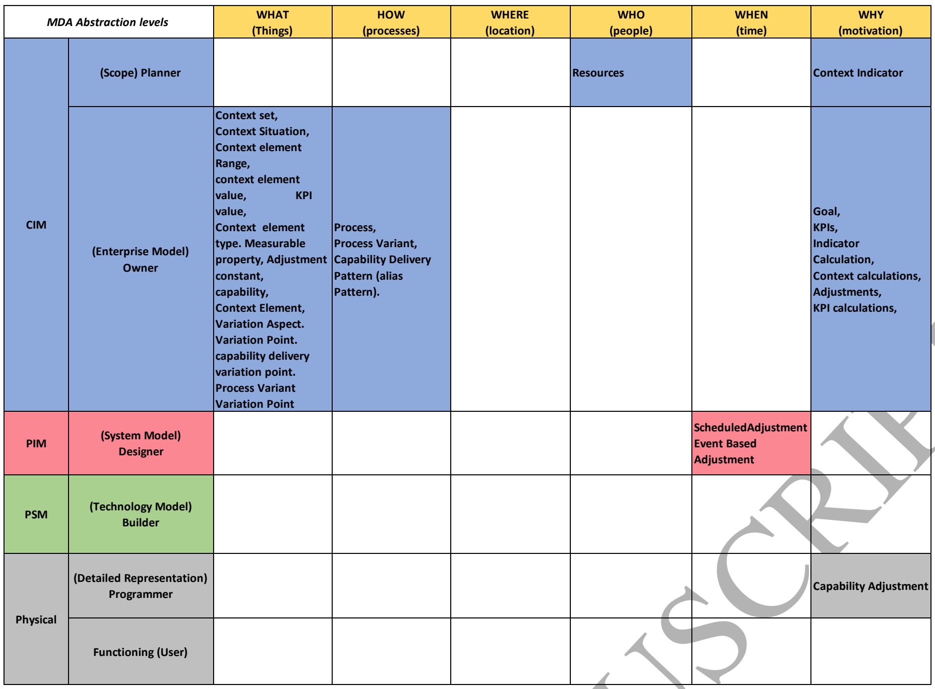
CDD is based on notational aspects for modelling enterprise concerns; its original specification is not model-driven compliance. However, in order to support the capability-driven development paradigm that is promoted in the CaaS EU project, CDD was managed by using a MDE approach which gives computational capacities to CDD such as tooling, modelling, repositories, and code generation. The CDD-MDE approach of CaaS was selected in this example by its explicit use of multiple languages for modelling enterprise concerns. In future works, we plan to use MMQEF in evaluating other modelling languages such as UML and BPMN that can eventually be applied at the enterprise level.

To start the taxonomic analysis, we use the available information of the modelling languages and their use; this comes from the specification of the CDD methodology (which speci-fies a metamodel for it), the explanation about each involved model (taken from the 4EM specification [44]), and examples of the CDD methodology as presented in [47]. In addition, in order to take advantage of the specific method for modelling business processes chosen in the CDD methodology[3], a previous analysis (reported in [48]) that uses the reference architecture for analyzing the BPMN notation was also used. Figure 5 gives a brief overview about the application of the MMQEF method in the CDD methodology.

One of the most classical reported ways to begin the taxonomic analysis is the association between diagrams and cells; this yields the classification shown in Figure 6. However, to perform the activities related in Table A.1, the information contained in the example diagrams of the 4EM is also considered. This information enables precise compliance of the element of the involved languages with the cells of the taxonomy. This

[3]In the CaaS project, processes are modelled using BPMN instead of the original proposal of 4EM [44].

ACCEPTED MANUSCRIPT



MDA Abstraction levels		WHAT (Things)	HOW (processes)	WHERE (location)	WHO (people)	WHEN (time)	WHY (motivation)
CIM	(Scope) Planner				Resources		Context Indicator
CIM	(Enterprise Model) Owner	Context set, Context Situation, Context element Range, context element value, KPI value, Context element type. Measurable property, Adjustment constant, capability, Context Element, Variation Aspect. Variation Point. capability delivery variation point. Process Variant Variation Point	Process, Process Variant, Capability Delivery Pattern (alias Pattern).				Goal, KPIs, Indicator Calculation, Context calculations, Adjustments, KPI calculations,
PIM	(System Model) Designer					ScheduledAdjustment Event Based Adjustment	
PSM	(Technology Model) Builder						
Physical	(Detailed Representation) Programmer						Capability Adjustment
Physical	Functioning (User)						

Figure 8: Taxonomic analysis of the metamodel of the CDD methodology.

compliance results in Figure 7, in which the first key quality issues appear. For example, there is an overlap in the *scope-planner/why* cell between the *goal* concept associated with the *goal model* and the *technical components and requirements model* diagrams.

A key issue detected from the obtained analysis in Figure 7 is the fulfilment of the *enterprise model - owner* row of the reference taxonomy by the association of the modelling elements belonging to this particular CDD approach to each column of this row. This constitutes a complete model from the *enterprise model- owner* row according to the *R6* rule of the taxonomy framework [16] (see Section 3.2). From this, it is possible to show how the CDD approach meets the organizational level regarding the scope of the CaaS project itself; however, this row contains other quality issues.

The suggested classification in the *enterprise model-owner/where* cell (extracted from ([48]) is quite questionable owing to the semantic considerations of the modelling elements of the BPMN related to this cell (pool and message flows). The proposed use of both concepts is far from the original definition of these concepts in the BPMN specification [49] (*participants and messages between them, respectively*). Thus, the *enterprise model- owner/where* cell should be empty owing to the lack of modelling support from the CDD approach for the *where* question, which is critical for managing specific implementation issues for an operational deployment in an enterprise. In this case, $ComP \rightarrow 0$. In addition, another quality issue originating from the previous BPMN classification proposal is in the *enterprise model- owner/who* cell, where the *workflow* concept is located, but it must be in the *enterprise model- owner/how* because this represents a business process. For the same cell, the *Organizational unit* element of the *Actors and resources model* diagram has no information about its physical location.

Figure 7 shows that the CDD methodology provides conceptual support for the PIM abstraction level and no support for the PSM and technical implementation levels. This is a critical issue considering the full operationalization of the results from the methodology in real computational platforms. Although the *NCC* metric (Section 3.5.2) demonstrates proper support of the approach for the CIM level (58.33% of CIM cells 7/12; 83.33% for *enterprise model owner* row, cells 5/6), no information about the following modelling elements is explicitly provided.

With regard to the activities shown in Table A.2, some issues were found regarding the *traceability* and *navigation between abstraction levels*. In the *enterprise model - owner/how* cell, the *actor* modelling element must have explicit relationships with *goals*, *rules* and *processes*, but these are not formally defined; instead, these are dependent on the analyst's criteria. Another quality issue is in the *system- what* cell, where there are no traceability relationships among the *business processes model*, the *actors and resources model*, and the *concepts model*, despite the explicit relations that they must have by the references proposed in the *requirement expressions*. In addition, *functional requirements* that are associated with this cell must be clearly defined with reference to the *concepts model*, but these relations are not explicitly defined.

Table A.2 also shows that there is evidence of preservation of a construct between the *identify the explicit traces that support the navigation between abstraction levels / viewpoints* activity and the *goal* concept of the *goal model*, which is in the *CIM*

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

scope planner / why and *CIM enterprise model owner / why* cells. For this concept, $CP = 1$. This is a result of the proposed operationalization for the *goal* concept by *business rules* or business processes. The justification of the *multiple viewpoint* is not provided by the methodology; however, it is derived from the use of the models for covering the enterprise concerns.

For the *Identify the capacities for model transformations* activities block of the MMQEF method (Table A.4), the capacities for transformation, mapping, and computational support of models are not covered by the CDD methodology, so these are delegated by the specific implementations of this CDD approach. For the *integration mechanisms* (Table A.5), the taxonomic analysis demonstrates that there is no full taxonomic independence between the *technical components and requirements model diagram* and the *goal diagram* owing to the closeness of the *goal* concept of both initiatives. This produces an overlap of the modelling tasks at the CIM level ($DoTx = 0$). The lack of *integration points* for each modelling language involved in the CDD methodology ($IP = 0$) generates cases in which concepts from the two models are presented in the same diagram without explicit support beyond the diagrammatic integration purposes (e.g., *goals* of the *goal model* and *rules* from the *business rules model* in an *enterprise model- owner/why* cell).

The classification reflects suitability evidence (activities shown in Table A.6). Most of the covered cells (77.77%) have at least two modelling languages that support them. In most cases, the analysis reflects that these languages are complementary regarding the semantic purpose (scope) of the cells (i.e., for each modelling language, its *ComP* value is proportional regarding the number of languages that support a specific cell). However, in the *CIM scope planner / why* cell, a suitability decision is required to choose the most appropriate language that models the *goal* and *problem* concepts; in this cell two different alternatives were detected for modelling both concepts. However, the CDD methodology does not indicate which language is the most appropriate modelling language for covering both concepts.

In Figure 8, the taxonomic analysis shows how most metaconcepts of the CDD methodology resolve concerns related to the data of the *context* and management of the *goals* at the *enterprise model-owner* level. For this reason, the taxonomic analysis differs from the previous analysis depicted in Figures 6 and 7. In addition, support for managing the variability of business services is detected at the same level. Normally this explicit consideration is part of the *context model*; however, its application is on points of processes. Therefore, the resulting classification is associated with the *enterprise model owner / how* cell for the semantic purposes of specifying variation scenarios.

3.7. Tool support

To support the application of the MMQEF quality evaluation method, we developed EMAT (*Eclipse Modelling Analytics Tool*). It is an Eclipse plugin for the Eclipse Modelling Framework project (one of the main model-driven technical environments). This plugin was developed to operationalize the classification process and its obtained data.

EMAT is formally supported by the *Formal Concept Analysis* (FCA) approach [50]. This FCA analysis considers the application of the seven rules of the reference taxonomy. The output of EMAT is not only a drawing of a connected graph, it is also a *conceptual model* of the semantic closeness among objects/attributes of modelling languages. The resulting lattices are rendered directly on the work area of EMF. However, lattices are not only a visual output with graphical information of the semantics, they are also *models* that contain data about the semantic closeness of modelling languages.

In order to operationalize MMQEF, we consider the reference taxonomy as a *formal context* where the *(object,attribute)* pairs are the incidences between the abstraction levels (rows) and the philosophical questions (columns), i.e., the metaconcepts of the taxonomy. These incidences are derived when the information of modelling languages is classified using the taxonomic structure. Thus, the FCA method processes the grammar constructs provided by the modelling languages that are involved in an IS development process (semantic constructs, diagrams, etc.), generating a connected graph or *concept lattice* as the output, where the relations between elements are described. The resulting lattice derives inferences about the application of modelling languages.

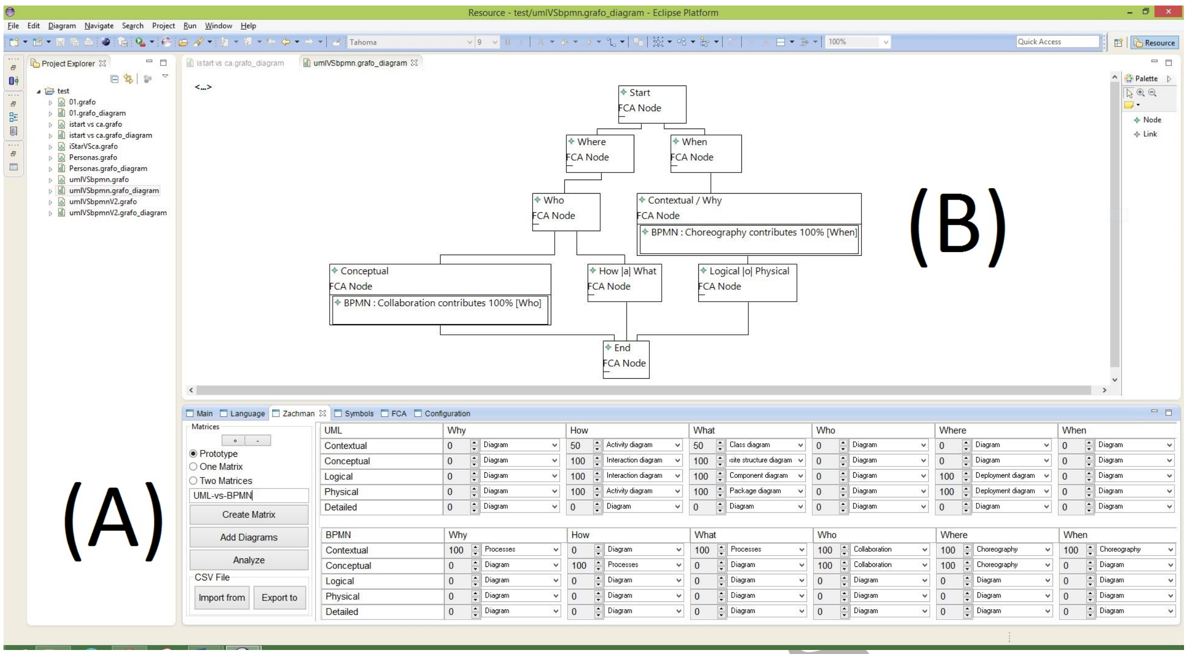
EMAT applies the notion of *object*, *attribute*, *concept*, *subconcept*, and *super-concept* to interpret an obtained lattice. This interpretation must be done top-down, from the start to end nodes, and taking into account that the attributes (i.e., the questions in the reference taxonomy) are those nearest to the start node, and the objects (i.e., the abstraction levels in the reference taxonomy) are nearest to the end node. An additional consideration is needed in the taxonomic analysis because EMAT provides a *completeness percentage* to indicate the degree of support of a modelling artifact for a specific concept. This percentage is depicted as internal nodes inside a conceptual node.

Figure 9 shows an example of a supported taxonomic analysis with the *classification* as input (Figure 9 - part A), and the semantic relations as output (Figure 9 - part B). Figures 10 and 11 depict the obtained results for the taxonomic analysis in the EMAT tool. Four conventions are required for understanding the FCA outputs of the EMAT tool; these are as follows:

- $/a/$ an *abstraction level* is associated only with the *viewpoint*.
- $/o/$ a *viewpoint* is associated only with an *abstraction level*.
- $|\ a\ |$ two *viewpoints* come together when they are related to two *abstraction levels*.
- $|\ o\ |$ two *abstraction levels* come together when they are related to the same *viewpoints*.

Figure 10 - part *A* presents the FCA lattice (or connected graph) for the classification previously shown in Figure 6. The $\{What/a/Logical\}$ node depicts the support that the *what* viewpoint offers for the *PIM - system model* level (or *logical* level in the original description of the taxonomy); this is because the

ACCEPTED MANUSCRIPT



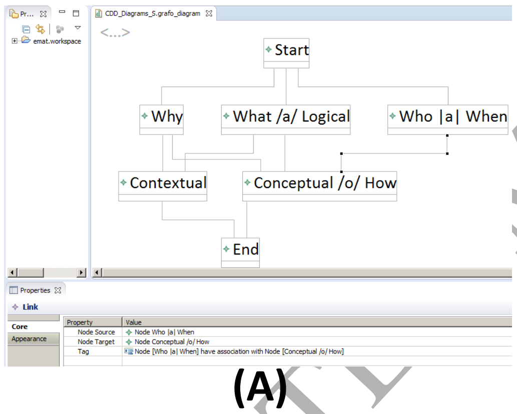
Main | Language | Zachman | Symbols | FCA | Configuration

Matrices
(● Prototype) One Matrix / Two Matrices
UML-vs-BPMN
Create Matrix | Add Diagrams | Analyze
CSV File: Import from | Export to

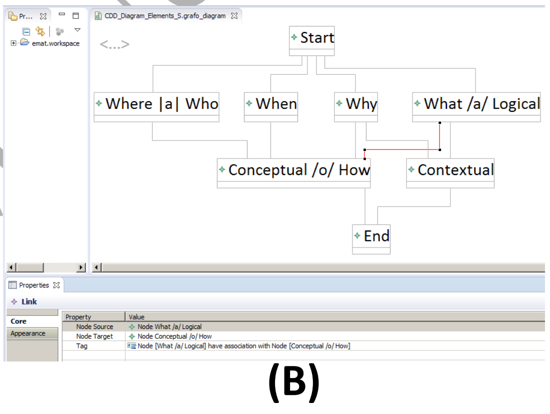
UML	Why		How		What		Who		Where		When	
Contextual	0	Diagram	50	Activity diagram	50	Class diagram	0	Diagram	0	Diagram	0	Diagram
Conceptual	0	Diagram	100	Interaction diagram	100	ısite structure diagram	0	Diagram	0	Diagram	0	Diagram
Logical	0	Diagram	100	Interaction diagram	100	Component diagram	0	Diagram	100	Deployment diagram	0	Diagram
Physical	0	Diagram	100	Activity diagram	100	Package diagram	0	Diagram	100	Deployment diagram	0	Diagram
Detailed	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram

BPMN	Why		How		What		Who		Where		When	
Contextual	100	Processes	0	Diagram	100	Processes	100	Collaboration	100	Choreography	100	Choreography
Conceptual	0	Diagram	100	Processes	0	Diagram	100	Collaboration	100	Choreography	0	Diagram
Logical	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram
Physical	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram
Detailed	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram

Figure 9: The EMAT tool for the MMQEF method.



Property	Value
Node Source	Node Who |a| When
Node Target	Node Conceptual /o/ How
Tag	Node [Who |a| When] have association with Node [Conceptual /o/ How]



Property	Value
Node Source	Node What /a/ Logical
Node Target	Node Conceptual /o/ How
Tag	Node [What /a/ Logical] have association with Node [Conceptual /o/ How]

Figure 10: The FCA lattice obtained from the classification shown in Figures 6 and 7.

what column is the only one that provides a diagram for these levels.

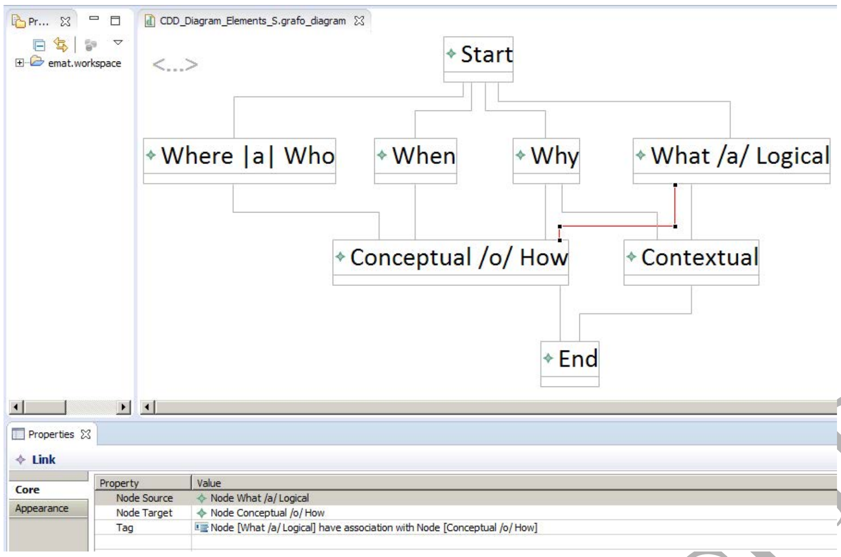
The {*Why*} node has relative independence because it is not sufficiently related to other viewpoints, which means that the diagrams that meet its associated cells (for the involved classifiers) are unique for this viewpoint. {*Why*} has a shared attribute only with the {*What/a/Logical*} viewpoint by the *Technical Components and Requirements Model (TCRM)* attribute. Thus, the {*Contextual*} node (or the *CIM - Scope* level) is represented as an independent node with relationships with the {*What/a/Logical*} and {*Why*} nodes because these are the only ones that offer support for modelling it. The {*Who* | *a* | *When*} node is generated for the tool to indicate that these viewpoints are related in the same abstraction level by sharing a common diagram. This node has semantic closeness with the {*Conceptual/o/How*} node (*CIM - Enterprise model* level) because it covers the same abstraction level but without sharing a specific diagram between the two nodes.

Figure 10 - part *B* presents the FCA lattice resulting from the taxonomic analysis on the modelling elements of the CDD methodology shown in Figure 7. Two direct associations were obtained in the {*Conceptual/o/How*} and {*What/a/Logical*} nodes because the modelling elements of these cells contain modelling elements exclusively, which means that these elements are not associated with other rows.

The {*Who* | *a* | *When*} node reflects an association at a conceptual level (*CIM - Enterprise model* level) between the *where* and *who* viewpoints because these both have a common abstraction level. The *why* and *when* nodes do not have a direct association with other abstraction levels, so these were processed as independent nodes.

The most relevant conclusion derived from the lattices is the

ACCEPTED MANUSCRIPT



Property	Value
Node Source	Node What /a/ Logical
Node Target	Node Conceptual /o/ How
Tag	Node [What /a/ Logical] have association with Node [Conceptual /o/ How]

Figure 11: The FCA lattice obtained from the classification shown in Figure 8.

full management of the *conceptual* row (the *CIM - Enterprise model* level) supported by the modelling approaches under analysis; this can be identified by the distance and number of input links to the {*Conceptual /o/ How*} node. Nodes with a combination of *abstraction levels* and *viewpoints* depict the existence of common modelling elements that derive a semantic closeness; depending on the semantic distance (the closeness between concepts of the modelling language under analysis) the FCA algorithm groups nodes or associates them by an explicit link. Figure 11 also shows this support, but in this case, the nodes differ from those presented in Figure 10 owing to the obtained classification of the metaelements of the CDD methodology.

EMAT provides computational support for analytics of MMQEF, which is complementary to the support that was previously described in Section 3.5.2. In this way, quality inferences (that are derived from the taxonomic analysis of MMQEF) are formally supported and potentially automatable by the application of some query analysis and/or formal methods. Quality inferences can be deduced and also automated from the model that is automatically generated by EMAT. Specific features and concerns regarding this tool will be described in further works. For now, our main intention is to relate the technological support provided for the MMQEF quality evaluation method, which is a critical issue because conventional modelling quality methods are difficult to implement (operationalize) in specific tools owing to their associated abstraction and theoretical background.

4. Preliminary trade-off analysis of the MMQEF method

4.1. Main implications of the MMQEF method

The most representative drawback of this evaluation method is its high coupling with the reference taxonomy. Even if the taxonomy is justified as a holistic description of the essential elements of an IS, its use is tied to the subjective criteria of the analyst to meet the expected conceptual elements of the taxonomy cells. Because of its neutral conception, the taxonomy does not have default modelling languages for its units. Instead, previous reports are identified that contain examples that attempt to clarify the expected models for each taxonomy unit. The MDA specification also does not prescribe default modelling languages for the MDA levels (it only promotes its UML support). Nevertheless, we propose taking advantage of these subjective criteria to promote reasoning on models, and thus justify the selected choices. The classification of modelling languages (and their associated artefacts) enables the verification of the sufficiency of a modelling approach to conceptually manage a specific IS concern.

4.1.1. Why this taxonomy?

Taxonomic analysis is a practical application of the classification as a fundamental mechanism for organizing knowledge. It addresses problems in conceptual modelling that are related to suitability issues (*which theory of concepts to use*)[51] and their derived quality phenomena. Classification provides the basis that helps to determine the *value* in an inference-based procedure [52].

Following the considerations presented in [52], the taxonomy is an explicit method for modelling semantics that describes the structure of knowledge (semantic domain) about things in the IS domain. The taxonomy defines a holistic description of an IS in an enterprise context, considering *computationally independent* aspects for their implementation in specific computerized platforms. It has an implicit mapping with the MDA levels [20], so this supports any phenomenon regarding the formulation and usage of models. By default, the taxonomy uses models to cover the essential features of the ISs.

The taxonomy provides an IS architecture description with the essential models required to cover all real concerns involved in an IS project (from organizational to computerized issues). The taxonomy focuses on the information that must be mana-

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

ged by the artefacts provided in modelling languages under consideration.

For other taxonomy proposals for model-driven initiatives (Section 2.1), the reference taxonomy has a more complete taxonomic scope because, by default, it considers viewpoints (and their resulting views) that are directly associated to business and technical levels where an IS will be deployed. Model operations (e.g., transformations) are the results of the interaction of models within IS levels.

Classification is the main purpose of the reference taxonomy independent of its derived commercial uses. The taxonomy defines *classifiers* (i.e., the essential elements for each column that are required to conceive and produce a model from an *abstraction level/viewpoint* combination). Classifiers are set according to the conceptual modelling foundations presented in [17, 18]. MDE covers all domains of IS architecture defined by the reference taxonomy [53].

Unlike most reported IS taxonomies, the reference taxonomy proposes classification using a set of essential elements that must be considered by an IS. This means that the classification of IS elements is set from an IS viewpoint. Few IS taxonomy proposals use modelling elements for the classification act itself; for example, [54] suggested *language* as a component of a taxonomy to classify methodologies for IS development.

4.2. Feasibility of the classification procedure for quality purposes

Currently, there are no reports about the explicit use of the taxonomy as a model evaluation tool that supports model analytics. However, there are some works that present the support of the taxonomy in modelling classification tasks. For example, [55] makes suggestions about modelling techniques for a medi-cal domain; these suggestions were made from the classifiers of the taxonomy. The authors also proposed the usage of individual perspectives of the taxonomy as *user interfaces* for a knowledge distribution system. In the analysis of the Zachman framework reported in [56], the authors suggested a set of modelling languages to populate each cell of the taxonomy structure according to the intention, design, and needs of the specific tasks for each cell. The reference taxonomy was used in [57] to classify the identified models for a specific domain under analysis. The reference taxonomy is often (and commonly) used to justify the scope of specific modelling initiatives regarding the scope of an IS holistic description.

4.3. MMQFE and other quality frameworks for MDE

The MMQEF framework takes advantage of taxonomic analysis to identify and evaluate quality issues for modelling elements, modelling languages, and models that define languages. The key premise is the use of an Information System (IS) reference architecture that proposes *classification* as the strategy for understanding all phenomena involved in an IS project. By default, *models* are the conceptual supports for these phenomena.

In this sense, MMQEF does not attempt to replace or create any previous method for evaluating quality in MDE. Conversely, MMQEF is a complementary approach that preserves and promotes previous quality frameworks because the classification act encourages quality considerations derived from sources such as semiotics and ontologies.

Current MDE quality frameworks and guidelines could be too abstract to be applicable for model-driven practitioners [58], most of whom are influenced by typical software development issues. MMQEF provides an operational framework, based on conceptual, methodological, and technological support, for performing procedures of quality evaluation that are aligned with previous quality principles (e.g., those prescribed in the SEQUAL framework [59], one of the most relevant quality initiatives for model-based and model-driven communities).

SEQUAL and MMQEF are complementary owing to their constructivist vision of modelling as a consequence of the interaction among the IS domain, the modelling languages, and the stakeholders involved with their associated knowledge. A classification process discovers the *true nature* of things, describing relationships among objects that should generate hypotheses [60]. Through the classification and the proposed analysis procedure, the MMQEF method implicitly considers the main components of SEQUAL and their quality types [59] (except for the *empirical quality* type, which can be better supported by works on concrete syntax and visual notation design such as [61, 62]). MMQEF also considers the categories of guidelines for the quality of modelling languages and the *pragmatic*, *social*, and *deontic* considerations for metamodels [63].

With regard to other model quality initiatives, MMQEF allows the *6C* quality goals defined in [64] to be identified and rationalized. Its methodological framework and the analytic support derived from it (discussed in Section 3.5.2) provide a conceptual infrastructure (from an IS reference architecture) for making precise argumentations directly from the modelling act over an IS.

Most current quality challenges for the model-driven paradigm come from previous IS frameworks such as FRISCO [65] (December 1996). It defines key aspects for the model-driven approach: the use of models (conceptual modelling), the definition of information systems, the use of information system denotations by representations (models), the definition of computerized information systems, and the *abstraction level zero* by the presence of processors (physical level).

FRISCO promotes the *communicative* factor as a key consequence of the use of models. FRISCO also suggests the need to harmonize modelling languages, presenting the *suitability* and *communicational* aspects for the modelling languages. Communication among stakeholders is critical for harmonization purposes because it allows them to discuss relevant quality issues from different views [14]. Therefore, FRISCO suggests relevant features for modelling languages (*expressiveness, arbitrariness*, and *suitability*).

These types of FRISCO challenges produce new concerns for model-driven practitioners. For example, suitability requires the usage of a variety of modelling languages, and communication requires that these languages must be compatible and harmonized. Suitability concludes that a diversity of modelling languages is required, so the differences among modelling languages (due to this diversity) are unjustified.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

MMQEF gives a methodological and technological framework that address these FRISCO challenges in modelling languages. The classification analysis enables identification of the purpose and use of the modelling languages involved in an IS project regarding the scope of cells. MMEQF also considers an explicit set of activities to rationalize the suitability of the languages, where decisions about this and harmonization are based on the coverage and completeness demonstrated in the reference taxonomy.

The presence of computational levels in the taxonomic analysis (abstraction level zero of FRISCO) binds the explicit computational support required for the modelling initiatives under evaluation; this ensures the coverage of IS concerns from the domain to computational levels with real deployment. In addition, the taxonomic analysis of MMQEF addresses and reflects the expressiveness provided by the languages, and it provides a practical approach to discuss the advantages/disadvantages of the modelling act on concerns in an IS. Communicational issues result from hypotheses generated in the clustering of modelling languages regarding the taxonomic structure and their use.

5. Conclusions

One of the main challenges of any quality evaluation method for the MDE paradigm is the demonstration of its feasibility for effectively addressing multiple issues that are derived by the central role of the models and their foundational modelling languages. While it is true that a single method does not cover all of the quality issues of model-driven projects, the methods can complement each other, similarly to the way that quality in traditional software contexts is managed, in which quality has multiple implications and evaluation procedures.

This work presents the main considerations of the MMQEF approach, which is a method for evaluating the quality of modelling languages and modelling artefacts that are used in the construction of Information Systems (IS) by classification procedures. We have described a scenario of applicability of the MMQEF method, in which multiple modelling languages were used for considering IS concerns in the context of business services. We have preliminarily presented a theoretical validation of our method, discussing the main implications that the method possesses (i.e., its high dependence on an IS reference architecture and its compliance with previous MDE quality initiatives).

With the specification of the MMQEF method, quality at the MDE level can be inferred from foundational descriptions and representations for IS. This is the starting point of a research program for quality evaluation in MDE that focuses on the systemic application of modelling languages in accordance with the foundational principles of the MDE paradigm and the involved viewpoints of IS projects. Further works are required to detail the evaluation procedures, expanding the main components of the method to obtain more granular components in a native language for their operationalization as a navigable process (such as a SPEM process with tasks, steps, artefacts, orientation, etc., or a SEMDM from ISO/IEC 24744:2014 [66]). Additional effort is required to demonstrate and discuss the formal support of the FCA approach to operationalize the MMQEF method (Section 3.7) in a popular native model-driven environment such as Eclipse EMF. We are currently validating the method with MDE practitioners; however, another work is projected to contrast our proposed method with previous MDE quality initiatives to find points of convergence and validate the method in model-driven scenarios and to develop a more detailed trade-off analysis of our approach.

Acknowledgements

Funding: This work was supported by COLCIENCIAS (Colombia) [grant 512, 2010]; the European Commision FP7 Project CaaS (611351); and ERDF structural funds.

References

[1] U. Frank, Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges, Software & Systems Modeling 13 (3) (2012) 941–962. doi:10.1007/s10270-012-0273-9.
URL http://dx.doi.org/10.1007/s10270-012-0273-9

[2] J. Krogstie, Model-Based Development and Evolution of Information Systems: A Quality Approach, Springer Publishing Company, Incorporated, 2012.

[3] Iso/iec/ieee systems and software engineering – architecture description, ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000) (2011) 1–46doi:10.1109/IEEESTD.2011.6129467.

[4] F. D. Giraldo, S. España, Ó. Pastor, W. J. Giraldo, Considerations about quality in model-driven engineering, Software Quality Journaldoi:10.1007/s11219-016-9350-6.
URL https://doi.org/10.1007/s11219-016-9350-6

[5] J. R. Romero, J. I. Jaén, A. Vallecillo, Realizing correspondences in multi-viewpoint specifications, in: 2009 IEEE International Enterprise Distributed Object Computing Conference, pp. 163–172.

[6] V. D. A. Visser, Eelco, W. Jos, Model-driven software evolution: A research agenda, in: In Proc. Int. Ws on Model-Driven Software Evolution held with the ECSMR07, 2007.

[7] J. Cabot, Is transitioning to mde revolutionary (for companies adopting it)? (jan 2013).
URL http://www.modeling-languages.com/is-transitioning-to-mde-revolutionary-for-companies-adopting-it/

[8] J. A. Zachman, A framework for information systems architecture, IBM Syst. J. 26 (3) (1987) 276–292. doi:10.1147/sj.263.0276.
URL http://dx.doi.org/10.1147/sj.263.0276

[9] A. Saghafi, Y. Wand, Do ontological guidelines improve understandability of conceptual models? a meta-analysis of empirical work, in: System Sciences (HICSS), 2014 47th Hawaii International Conference on, 2014, pp. 4609–4618. doi:10.1109/HICSS.2014.567.

[10] R. Weber, Y. Wand, An ontological model of an information system, IEEE Transactions on Software Engineering 16 (1990) 1282–1292. doi:10.1109/32.60316.
URL doi.ieeecomputersociety.org/10.1109/32.60316

[11] Y. Wand, R. Weber, On the deep structure of information systems, Information Systems Journal 5 (3) (1995) 203–223. doi:10.1111/j.1365-2575.1995.tb00108.x.
URL http://dx.doi.org/10.1111/j.1365-2575.1995.tb00108.x

[12] G. Guizzardi, G. Wagner, A unified foundational ontology and some applications of it in business modeling., in: CAiSE Workshops (3), 2004, pp. 129–143.

[13] G. Guizzardi, Ontological foundations for structural conceptual models, CTIT, Centre for Telematics and Information Technology, 2005.

[14] V. A. Shekhovtsov, H. C. Mayr, C. Kop, Chapter 3 - harmonizing the quality view of stakeholders, in: I. M. B. E. R. Stal (Ed.), Relating System Quality and Software Architecture, Morgan Kaufmann, Boston, 2014, pp. 41 – 73. doi:http://dx.doi.org/10.1016/B978-0-12-

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

417009-4.00003-X.
URL http://www.sciencedirect.com/science/article/pii/B-\978012417009400003X

[15] M. Rosemann, P. Green, Integrating multi-perspective views into ontological analysis, in: Proceedings of the Twenty First International Conference on Information Systems, ICIS '00, Association for Information Systems, Atlanta, GA, USA, 2000, pp. 618–627.
URL http://dl.acm.org/citation.cfm?id=359640.359915

[16] J. F. Sowa, J. A. Zachman, Extending and formalizing the framework for information systems architecture, IBM Syst. J. 31 (3) (1992) 590–616. doi:10.1147/sj.313.0590.
URL http://dx.doi.org/10.1147/sj.313.0590

[17] B. Thalheim, The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling, in: D. W. Embley, B. Thalheim (Eds.), Handbook of Conceptual Modeling, Springer Berlin Heidelberg, 2011, pp. 543–577. doi:10.1007/978-3-642-15865-0_17.
URL http://dx.doi.org/10.1007/978-3-642-15865-0_17

[18] B. Thalheim, Syntax, semantics and pragmatics of conceptual modelling, in: G. Bouma, A. Ittoo, E. Mtais, H. Wortmann (Eds.), Natural Language Processing and Information Systems, Vol. 7337 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 1–10. doi:10.1007/978-3-642-31178-9_1.
URL http://dx.doi.org/10.1007/978-3-642-31178-9_1

[19] OMG, MDA Guide Version 1.0.1, http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf (Jun. 2003).
URL http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

[20] D. S. Frankel, P. Harmon, J. Mukerji, J. Odell, M. Owen, P. Rivitt, M. Rosen, R. M. Soley, The Zachman Framework and the OMG's Model Driven Architecture (Sep. 2003).
URL http://petros.omg.org/mda/mda_files/09-03-WP_Mapping_MDA_to_Zachman_Framework1.pdf

[21] A. Fatolahi, S. S. Somé, T. C. Lethbridge, Managing Worldwide Operations and Communications with Information Technology, 1st Edition, Information Resources Management Association, 2007, Ch. Enterprise Architecture using the Zachman Framework: A Model Driven Approach.

[22] S. S. Ostadzadeh, F. S. Aliee, S. A. Ostadzadeh, A Method for Consistent Modeling of Zachman Framework Cells, Springer Netherlands, Dordrecht, 2007, pp. 375–380. doi:10.1007/978-1-4020-6264-3_65.
URL https://doi.org/10.1007/978-1-4020-6264-3_65

[23] J. A. Zachman, Excerpted from the zachman framework: A primer for enterprise engineering and manufacturing (omg-brwg rfi response assessment) (2003).
URL https://pdfs.semanticscholar.org/bbd4/9c62bef4a6c538ccde1396d60860fad0a056.pdf

[24] OMG, MDA Guide revision 2.0 (Jun. 2014).
URL http://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf

[25] D. Harel, B. Rumpe, Meaningful modeling: What's the semantics of "semantics"?, Computer 37 (10) (2004) 64–72. doi:10.1109/MC.2004.172.
URL http://dx.doi.org/10.1109/MC.2004.172

[26] J. E. Burge, J. M. Carroll, R. McCall, I. Mistrk, Rationale-Based Software Engineering, 1st Edition, Springer Publishing Company, Incorporated, 2008.

[27] Object and Reference Model Subcommittee of the Architecture Board, A Proposal for an MDA Foundation Model, ormsc/10-09-06, Tech. rep., Object Management Group (Sep. 2010).
URL http://www.omg.org/cgi-bin/doc?ormsc/10-09-06.pdf

[28] R. Smith, On the value of a taxonomy in modeling, in: A. Tolk (Ed.), Ontology, Epistemology, and Teleology for Modeling and Simulation, Vol. 44 of Intelligent Systems Reference Library, Springer Berlin Heidelberg, 2013, pp. 241–254. doi:10.1007/978-3-642-31140-6_13.
URL http://dx.doi.org/10.1007/978-3-642-31140-6_13

[29] K. Sandkuhl, J. Stirna, Template for the documentation of method components in caas, Tech. rep., Capability as a Service in digital enterprises - Collaborative Project Number 611351.
URL http://caas-project.eu/

[30] G. Goldkuhl, M. Lind, U. Seigerroth, Method integration: the need for a learning perspective, Software, IEE Proceedings - 145 (4) (1998) 113–118. doi:10.1049/ip-sen:19982197.

[31] M. Bjekovi, H. Proper, J.-S. Sottet, Enterprise modelling languages, in: B. Shishkov (Ed.), Business Modeling and Software Design, Vol. 173 of Lecture Notes in Business Information Processing, Springer International Publishing, 2014, pp. 1–23. doi:10.1007/978-3-319-06671-4_1.
URL http://dx.doi.org/10.1007/978-3-319-06671-4_1

[32] S. Brinkkemper, M. Saeki, F. Harmsen, Meta-modelling based assembly techniques for situational method engineering, Information Systems 24 (3) (1999) 209 – 228, 10th International Conference on Advanced Information Systems Engineering. doi:http://dx.doi.org/10.1016/S0306-4379(99)00016-2.
URL http://www.sciencedirect.com/science/article/pii/-\S0306437999000162

[33] I. v. d. Weerd, S. Brinkkemper, Meta-modeling for situational analysis and design methods, in: Handbook of Research on Modern Systems Analysis and Design Technologies and Applications, IGI Global, 2009, pp. 38–58.

[34] N. Guarino, C. A. Welty, A formal ontology of properties, in: Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management, EKAW '00, Springer-Verlag, London, UK, UK, 2000, pp. 97–112.
URL http://dl.acm.org/citation.cfm?id=645361.650850

[35] N. Malik, Why the zachman framework is not an ontology @ONLINE (Dec. 2009).
URL http://blogs.msdn.com/b/nickmalik/archive/2009/12/-\18/why-the-zachman-framework-is-not-an-ontology.aspx

[36] A. Kleppe, Software Language Engineering: Creating Domain-Specific Languages Using Metamodels, 1st Edition, Addison-Wesley Professional, 2008.

[37] C. Gurr, Combining semantic and cognitive accounts of diagrams, in: M. Anderson, B. Meyer, P. Olivier (Eds.), Diagrammatic Representation and Reasoning, Springer London, 2002, pp. 125–140. doi:10.1007/978-1-4471-0109-3_7.
URL http://dx.doi.org/10.1007/978-1-4471-0109-3_7

[38] D. L. Moody, Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions, Data & Knowledge Engineering 55 (3) (2005) 243–276.

[39] V. R. Basili, G. Caldiera, H. D. Rombach, The goal question metric approach, in: Encyclopedia of Software Engineering, Wiley, 1994.

[40] J. Stirna, Caas capability as a service for digital enterprises. fp 7 ict programme collaborative project no: 611351 (2013).
URL http://caas-project.eu/

[41] J. C. Egido, T. González, R. Juanes, C. Llorca, J. Grabis, J. Stirna, J. Zdravkovic, Capability as a service in digital enterprises collaborative project number 611351. deliverable 1.3: Vision of the cdd methodology, Tech. rep. (jan 2014).

[42] S. BrziÅa, S. España, J. Grabis, M. Henkel, L. Jokste, J. Kampars, H. Ko, K. Sandkuhl, J. Stirna, F. Valverde, J. Zdravkovic, Capability as a service in digital enterprises collaborative project number 611351. deliverable 5.2: The initial version of capability driven development methodology, Tech. rep. (feb 2015).
URL 10.13140/RG.2.1.2399.4965

[43] P. A. Bubenko, J. A., J. Stirna, User guide of the knowledge management approach using enterprise knowledge patterns, deliverable d3. ist programme project hypermedia and pattern based knowledge management for smart organisations, project no. ist- 2000-28401, Tech. rep. (oct 2001).

[44] K. Sandkuhl, J. Stirna, A. Persson, M. Wiotzki, Enterprise Modeling: Tackling Business Challenges with the 4EM Method, Springer Publishing Company, Incorporated, 2014.

[45] C. L. Quevedo, S. España, M. E. García, A. G. Hita, T. G. Cardona, J. Grabis, M. Henkel, R. J. Pascual, L. Jokste, F. V. Giromé, Capability as a service in digital enterprises collaborative project number 611351 deliverable 4.2 capability models for soa technological platforms, Tech. rep. (jan 2015).

[46] S. España, T. González, J. Grabis, L. Jokste, R. Juanes, F. Valverde, Advanced Information Systems Engineering Workshops: CAiSE 2014 International Workshops, Thessaloniki, Greece, June 16-20, 2014. Proceedings, Springer International Publishing, Cham, 2014, Ch. Capability-Driven Development of a SOA Platform: A Case Study, pp. 100–111. doi:10.1007/978-3-319-07869-4_9.
URL http://dx.doi.org/10.1007/978-3-319-07869-4_9

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

[47] P. R. Centre, Capability delivery in action: Showcasing the cdd approach (2015).
URL http://pros2.webs.upv.es/index.php/es/yespoem/enterprise-models-example

[48] L. Zhao, K. Letsholo, E.-V. Chioasca, S. Sampaio, P. Sampaio, Can business process modeling bridge the gap between business and information systems?, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12, ACM, New York, NY, USA, 2012, pp. 1723–1724. doi:10.1145/2245276.2232054.
URL http://doi.acm.org/10.1145/2245276.2232054

[49] O. M. G. (OMG), Business process model and notation (bpmn) version 2.0, Tech. rep. (jan 2011).
URL http://www.omg.org/spec/BPMN/2.0/PDF/

[50] U. Priss, Formal concept analysis in information science, Annual Rev. Info. Sci & Technol. 40 (1) (2006) 521–543. doi:10.1002/aris.v40:1.
URL http://dx.doi.org/10.1002/aris.v40:1

[51] Y. Wand, D. E. Monarchi, J. Parsons, C. C. Woo, Theoretical foundations for conceptual modelling in information systems development, Decision Support Systems 15 (4) (1995) 285 – 304. doi:http://dx.doi.org/10.1016/0167-9236(94)00043-6.
URL http://www.sciencedirect.com/science/article/pii/-\0167923694000436

[52] J. Parsons, Y. Wand, Using cognitive principles to guide classification in information systems modeling, MIS Quarterly 32 (4) (2008) pp. 839–868.
URL http://www.jstor.org/stable/25148874

[53] A. Brossard, M. Abed, C. Kolski, Taking context into account in conceptual models using a model driven engineering approach, Information and Software Technology 53 (12) (2011) 1349 – 1369. doi:http://dx.doi.org/10.1016/j.infsof.2011.06.011.
URL http://www.sciencedirect.com/science/article/pii/-\S0950584911001583

[54] K. Lyytinen, Critical issues in information systems research, John Wiley &amp; Sons, Inc., New York, NY, USA, 1987, Ch. A Taxonomic Perspective of Information Systems Development: Theoretical Constructs and Recommendations, pp. 3–41.
URL http://dl.acm.org/citation.cfm?id=54905.54906

[55] J. Kingston, A. Macintosh, Knowledge management through multi-perspective modelling: representing and distributing organizational memory, Knowledge-Based Systems 13 (23) (2000) 121 – 131. doi:http://dx.doi.org/10.1016/S0950-7051(00)00053-8.
URL http://www.sciencedirect.com/science/article/pii/-\S0950705100000538

[56] O. Noran, An analysis of the zachman framework for enterprise architecture from the {GERAM} perspective, Annual Reviews in Control 27 (2) (2003) 163 – 183. doi:http://dx.doi.org/10.1016/j.arcontrol.2003.09.002.
URL http://www.sciencedirect.com/science/article/pii/-\S1367578803000269

[57] Y. Liao, M. Lezoche, H. Panetto, N. Boudjlida, E. Rocha Loures, Semantic Annotation for Knowledge Explicitation in a Product Lifecycle Management Context: a Survey, Computers in Industry 71 (2015) 24–34. doi:10.1016/j.compind.2015.03.005.
URL https://hal.archives-ouvertes.fr/hal-01123854

[58] J. Mendling, H. A. Reijers, W. M. P. van der Aalst, Seven process modeling guidelines (7pmg), Inf. Softw. Technol. 52 (2) (2010) 127–136. doi:10.1016/j.infsof.2009.08.004.
URL http://dx.doi.org/10.1016/j.infsof.2009.08.004

[59] J. Krogstie, Model-Based Development and Evolution of Information Systems: A Quality Approach, Springer London, London, 2012, Ch. Quality of Models, pp. 205–247. doi:10.1007/978-1-4471-2936-3_4.
URL http://dx.doi.org/10.1007/978-1-4471-2936-3_4

[60] R. R. Sokal, Classification: Purposes, principles, progress, prospects, Science 185 (4157) (1974) 1115–1123.
URL http://www.jstor.org/stable/1738359

[61] T. Green, M. Petre, Usability analysis of visual programming environments: A cognitive dimensions framework, Journal of Visual Languages & Computing 7 (2) (1996) 131 – 174. doi:http://dx.doi.org/10.1006/jvlc.1996.0009.
URL http://www.sciencedirect.com/science/article/pii/-\S1045926X96900099

[62] D. Moody, The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering, IEEE Trans. Softw. Eng. 35 (6) (2009) 756–779. doi:10.1109/TSE.2009.67.
URL http://dx.doi.org/10.1109/TSE.2009.67

[63] J. Krogstie, Model-Based Development and Evolution of Information Systems: A Quality Approach, Springer London, London, 2012, Ch. Quality of Modelling Languages, pp. 249–280. doi:10.1007/978-1-4471-2936-3_5.
URL http://dx.doi.org/10.1007/978-1-4471-2936-3_5

[64] P. Mohagheghi, V. Dehlen, T. Neple, Definitions and approaches to model quality in model-based software development - a review of literature, Inf. Softw. Technol. 51 (12) (2009) 1646–1669. doi:10.1016/j.infsof.2009.04.004.
URL http://dx.doi.org/10.1016/j.infsof.2009.04.004

[65] E. D. Falkenberg, W. Hesse, P. Lindgreen, B. E. Nilsson, J. L. H. Oei, C. Rolland, R. K. Stamper, F. J. M. V. Assche, A. A. Verrijn-Stuart, K. Voss, Frisco: A framework of information system concepts, Tech. rep., The IFIP WG 8. 1 Task Group FRISCO (1996).
URL http://cs-exhibitions.uni-klu.ac.at/index.php?id=445

[66] I. O. for Standardization/International Electrotechnical Commission, et al., Iso/iec 24744:2014, Software Engineering-Metamodel for Development Methodologies.

[67] D. Harel, B. Rumpe, Modeling languages: Syntax, semantics and all that stuff, part i: The basic stuff, Tech. rep., Jerusalem, Israel, Israel (2000).

[68] P. Adriaans, Information, in: E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, fall 2013 Edition, 2013.

[69] A. R. da Silva, Model-driven engineering: A survey supported by the unified conceptual model, Computer Languages, Systems & Structures 43 (2015) 139 – 155. doi:http://dx.doi.org/10.1016/j.cl.2015.06.001.
URL http://www.sciencedirect.com/science/article/pii/-\S1477842415000408

[70] omg, Meta Object Facility (MOF) Core Specification Version 2.5 (2015).
URL http://www.omg.org/spec/MOF/2.5/

Appendix A. The process-delivery diagram (PDD) specification for the MMQEF method

Continuing with the specification started in Section 3, in this appendix, we show the associated activities and concepts required for the MMEQF method in more detail. Tables A.1 to A.6 provide a more thorough description of the activities associated to the evaluation of quality of modelling languages through the reference taxonomy. Following the PDD approach, each activity block is mapped to a *concept diagram*, which contains the key elements involved in the taxonomic analysis. Concepts depicted in gray describe terms that are extracted from previous foundational models, such as the MDA foundation model [27] and the ISO 42010 [3] conceptual models.

In addition, Tables A.7 and A.8 describe the concepts involved in the taxonomic evaluation procedure. The main purpose of these tables is to show that the main concepts are from relevant referents for the MDE conceptualization. Few concepts are required to understand the full applicability of the taxonomic analysis.

ACCEPTED MANUSCRIPT

Activity	Sub-activity	Description
Determine the organization of the modelling language	Associate the elements of the modelling language under analysis to taxonomy cells	Each one of the input elements of a modelling language (either REPRESENTATIONs or ABSTRACT SYNTAX) are located in a cell or a set of CELLs based on the closeness of the element with the purpose of the CELLs (i.e., the FOUNDATIONAL CONSTRUCT of the CELLs with the associated ABSTRACTION LEVEL.)
	Relate the information extracted from diagrams to the essential model of the analyzed cell	Key concepts associated to the INFORMATION depicted by REPRESENTATIONs that belong to modelling languages are contrasted regarding the scope of the essential model that governs the taxonomic unit (CELL or set of CELLs). Key concepts could be either conceptual entities or operations.
	Resolve the compliance of the modelling language elements with DSL cell constructs	A rationale about why the information of the element (REPRESENTATION, ABSTRACT SYNTAX) meets the essential model that governs the specific CELL(s) involved.

Table A.1: Description of activities related to the *Determine the organization of the modelling language* block.

Activity	Sub-activity	Description
Identify the explicit traces that support the navigation between abstraction levels / viewpoints	Find the traceability mechanism	These are the capacities of the language for supporting the incremental evolution of FOUNDATIONAL CONSTRUCTs inside a specific column (VIEWPOINT), checking how the FOUNDATIONAL CONSTRUCT of the VIEWPOINT is preserved when it passes through all of the ABSTRACTION LEVELs that the LANGUAGE supports. For example, in the *What* VIEWPOINT (data FOUNDATIONAL CONSTRUCT), a traceability link is from the *Domain model conceptual model ER entities* tables in a SQL engine. Thus, TRACEs are from MODELs belonging to the same VIEWPOINT and some ABSTRACTION LEVELs. If the TRACEs are from PIM-PSM levels, these are a specification of a MAPPING.
	Identify the support for each abstraction level supported	Key concepts associated to the INFORMATION depicted by REPRESENTATIONs that belong to modelling languages are contrasted regarding the scope of the essential model that governs the unit. For each of the modelling artifacts under analysis, the TRACEs and MAPPINGS relations are checked in order to determine the support that the modelling artifacts provide to the MDA levels (CIM, PIM, PSM). This activity checks the relationships between elements of a MODELLING LANGUAGE that supports multiple ABSTRACTION LEVELs in order to determine the explicit MAPPING between MODELs of different levels.

Table A.2: Description of activities related to the *Identify the explicit traces that support the navigation between abstraction levels / viewpoints* block (Part I).

Activity	Sub-activity	Description
Identify the explicit traces that support the navigation between abstraction levels / viewpoints	Identify explicit mechanisms for taxonomic metamodel preservation	This verifies how organizational-domain and system concerns are traced until specific technical implementations. In this way, the execution of model transformations are in accordance with the semantic domain where the modelling act occurs. It helps detect whether the reasoning about the relations are a consequence of progressive preservation of semantic constructs (that is added to incremental INFORMATION of the respective level in a top-down path). In addition, the relation could be the result of semantic changes introduced by considering at least two different viewpoints (e.g., looking for the support of a modelling artifact in the same row or ABSTRACTION LEVEL of the REFERENCE TAXONOMY).
	Reason about the justification of multi-viewpoint supported relationships	When the resulting organization of model elements of the REFERENCE TAXONOMY indicates some relationships between INFORMATION in different viewpoints (two or more), supported by a derivation rationale, a justification about why this derivation is compliant with the FOUNDATIONAL CONSTRUCTs of the involved CELLs is required. This analysis makes it possible to identify what the contribution is for making a single model of each ABSTRACTION LEVEL from the classified MODEL ELEMENT.

Table A.3: Description of activities related to the *Identify the explicit traces that support the navigation between abstraction levels / viewpoints* block (Part II).

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Activity	Sub-activity	Description
Identify the capacities for model transformations	Find the support for transformation specifications	This activity promotes the specification of TRANSFORMATION SPECIFICATION MODELs from the semantic closeness of concepts belonging to the ABSTRACT SYNTAX of the involved MODELs or modelling artifacts. The semantic closeness is the explicit association of the involved concepts with the associated FOUNDATIONAL CONCEPTs of the CLASSIFIERs from CELLs.
	Determine the support for mappings	It verifies if MODEL ELEMENTs classified at the System level of the REFERENCE TAXONOMY can generate MODEL ELEMENTs at the Technology level of the same VIEWPOINT of the taxonomy.
	Define the explicit computational support of the modelling language	According to the TRACE mechanisms provided by the MODELLING LANGUAGE, this activity checks them in order to identify the respective computational support (code, logical or physical computational artifact) for conceptual MODELs from higher levels.

Table A.4: Description of activities related to the *Identify the capacities for model transformations* block.

Activity	Sub-activity	Description
Find the mechanism for integration	Identify explicit viewpoint integration support	This reviews the capabilities offered by the MODELLING LANGUAGE for integrating it with other languages and verifies that a modelling artifact focuses only on an IS CONCERN so that new related concerns will be managed by other more suitable modelling mechanisms.
	Establish the degree of taxonomic independence	This verifies that the proposed classification for a MODELLING LANGUAGE, using the REFERENCE TAXONOMY, is independent of previous classifications formulated for the same language through its analysis on its CONSTRUCTs or INFORMATION extracted from REPRESENTATION. It establishes the clearly differentiating features for the language, which serves to justify its applicability over a set of IS CONCERNs.
	Define the coverage for each abstraction level	This determines how much information can be captured by the MODELLING LANGUAGE for an IS project. The amount of information is calculated from the number of CELLs covered by the MODELLING LANGUAGE through its classified elements.

Table A.5: Description of activities related to the *Find the mechanism for integration* block.

Activity	Sub-activity	Description
Define suitability issues	Define semantic commonality between languages	For all the MODELLING LANGUAGEs that share a CELL in the taxonomic analysis, the specific elements that produce the classification in the involved CELLs are identified.
	Select modelling alternatives	This is a decision about the best alternatives for modelling an IS CONCERN based on the coverage supported by the MODELLING LANGUAGEs. Prioritization of decisions are based on the total support for a CONCERN managed by a CELL or the coverage of CELLs.

Table A.6: Description of activities related to the *Define suitability issues* block.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Concept	Description
VIEWPOINT	From [3, 27], it represents the criteria and the set of conventions used for formulating views and especially for framing CONCERNS.
CONCERN	From [3, 27], this is the interest in a system of a stakeholder.
ABSTRACTION LEVEL	It refers to a restriction proposed to model and manage specific phenomena on an IS. The MDA architecture specification defines some abstract hierarchical level (CIM, Computational-Independent Model; PIM, Platform-Independent Model; PSM, Platform-Specific Model Technical Implementation), which can frame the models used in a model-driven project.
FOUNDATIONAL CONSTRUCT	Enumeration that represents each of the single models associated to each column of the taxonomy: What (thing), How (process), Why (purpose), Where (node), When (event), Who (people).
CLASSIFIER	It refers to the logical combination (crossing) of the FOUNDATIONAL CONSTRUCTs of each column of the taxonomy, with the ABSTRACTION LEVELs defined in the MDA reference architecture applied in the taxonomy.
CELL	Each one of the graphical elements that compose the REFERENCE TAXONOMY, which allows elements to be classified according to the CLASSIFIERs that these contain.
REFERENCE TAXONOMY	An ARCHITECTURE DESCRIPTION of an IS, which considers all essential elements that conform this system, considering them from organizational to technical implementation ABSTRACTION LEVELs.
ARCHITECTURE DESCRIPTION	According to [3], it is a word product used to express (depict) an architecture.
TRANSFORMATION SPECIFICATION	According to [27], it is a model that defines how different elements will relate to each other. These elements belong to models or artifacts of the same system at different levels of refinement.
MAPPING	According to [27], it is a model that provides specifications for transformation of a PIM ABSTRACTION LEVEL into a PSM ABSTRACTION LEVEL for a specific platform. The platform model will determine the nature of the mapping.
TRACE	According to [27], it is the set of INFORMATION that defines how a CONCERN is preserved throughout its crossing of all the ABSTRACTION LEVELs involved in an IS project (from a CIM to Technical Implementation levels).

Table A.7: Concept table for the MMQEF method in PDD convention (I).

Concept	Description
MODEL	From [3, 27], a model can be anything: a concept or a work product. A model is valid if it helps to answer questions about a system under consideration.
INFORMATION	According to [67], INFORMATION is the result of an *interpretation* process that assigns a meaning to each piece of *data*. Thus, the data is a syntactic representation of INFORMATION. The relation between INFORMATION and data is implicit. [68] defines the INFORMATION as an abstract mass-noun used to denote any amount of data, code, or text that is stored, sent, received, or manipulated in any medium.
MODELLING LANGUAGE	We coincide with the definition presented in [69], where the modelling language is a set of all possible MODELs that are conformant with an ABSTRACT SYNTAX, represented by one or more CONCRETE SYNTAX, and that satisfy a given semantic.
ABSTRACT SYNTAX	A *metamodel* with all the concepts identified for a MODELLING LANGUAGE at the meta-domain level [69]. It identifies the concepts, abstractions, and relations underlying the application domain.
GRAMMAR	A specific technique for the definition of an ABSTRACT SYNTAX for textual languages (textual DSLs) and even natural languages.
METAMODEL	A specific technique for the definition of an ABSTRACT SYNTAX for modelling languages, using an UML profile mechanism of the class diagram that is compliant with the MOF language [70] (or its variations).
CONCRETE SYNTAX	According to [69], it is the notation, or the way users of a modelling language will learn and will use it, either by reading or by writing and designing the models.
GUIDELINE	Continuing with the definition presented in [69], it is a consequence of the *pragmatics* of a MODELLING LANGUAGE, which helps and guides how to use it in the most appropriate way.
REPRESENTATION	It refers to the depiction employed by a modelling language for representing a (or a set of) MODEL(s) that expresses some concern in an IS.
DIAGRAM	A REPRESENTATION that uses graphical symbols, generally based on nodes (in a connected lattice). These have lines that represent relationships with each other.
TEXTUAL	A REPRESENTATION that uses textual symbols for modelling specific concerns (generally textual DSLs).

Table A.8: Concept table for the MMQEF method in PDD convention (II).

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

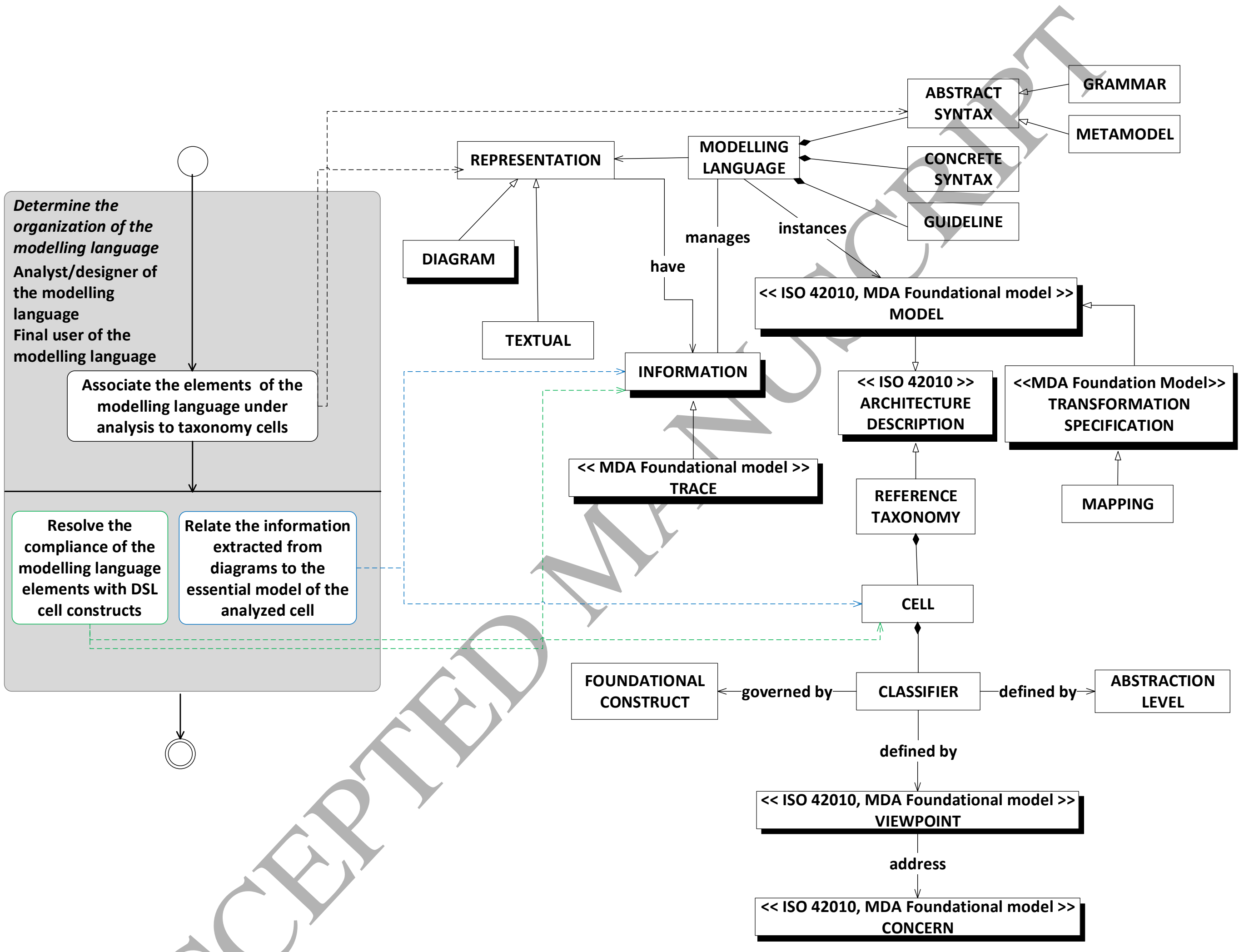

Determine the organization of the modelling language
Analyst/designer of the modelling language
Final user of the modelling language
Associate the elements of the modelling language under analysis to taxonomy cells
Resolve the compliance of the modelling language elements with DSL cell constructs
Relate the information extracted from diagrams to the essential model of the analyzed cell
ABSTRACT SYNTAX
GRAMMAR
METAMODEL
MODELLING LANGUAGE
REPRESENTATION
CONCRETE SYNTAX
GUIDELINE
DIAGRAM
TEXTUAL
manages
instances
have
<< ISO 42010, MDA Foundational model >> MODEL
INFORMATION
<< ISO 42010 >> ARCHITECTURE DESCRIPTION
<<MDA Foundation Model>> TRANSFORMATION SPECIFICATION
<< MDA Foundational model >> TRACE
REFERENCE TAXONOMY
MAPPING
CELL
FOUNDATIONAL CONSTRUCT
governed by
CLASSIFIER
defined by
ABSTRACTION LEVEL
defined by
<< ISO 42010, MDA Foundational model >> VIEWPOINT
address
<< ISO 42010, MDA Foundational model >> CONCERN

Figure A.12: Association of activities of the *Determine the organization of the modelling language* block with concepts of the MMQEF metamodel.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

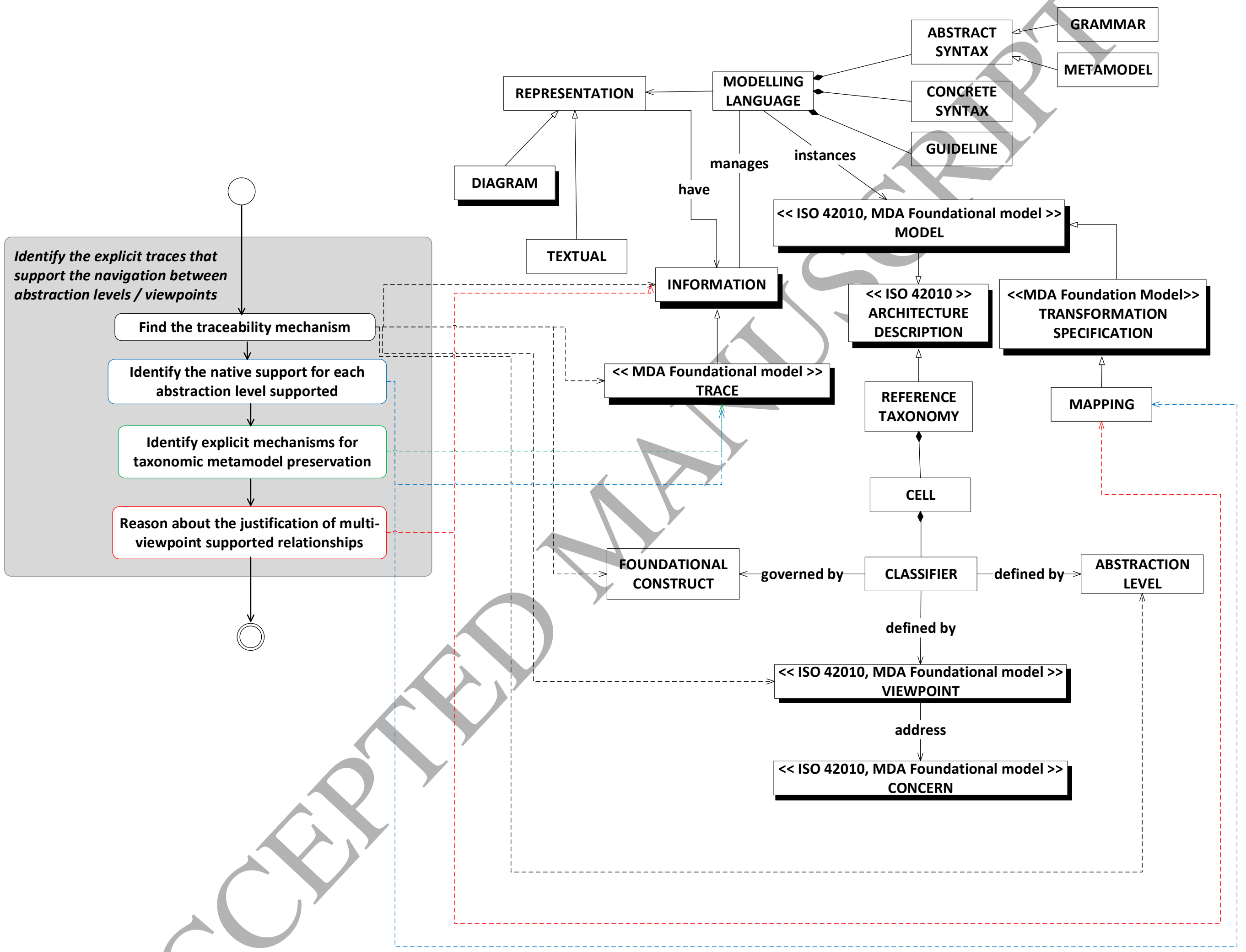
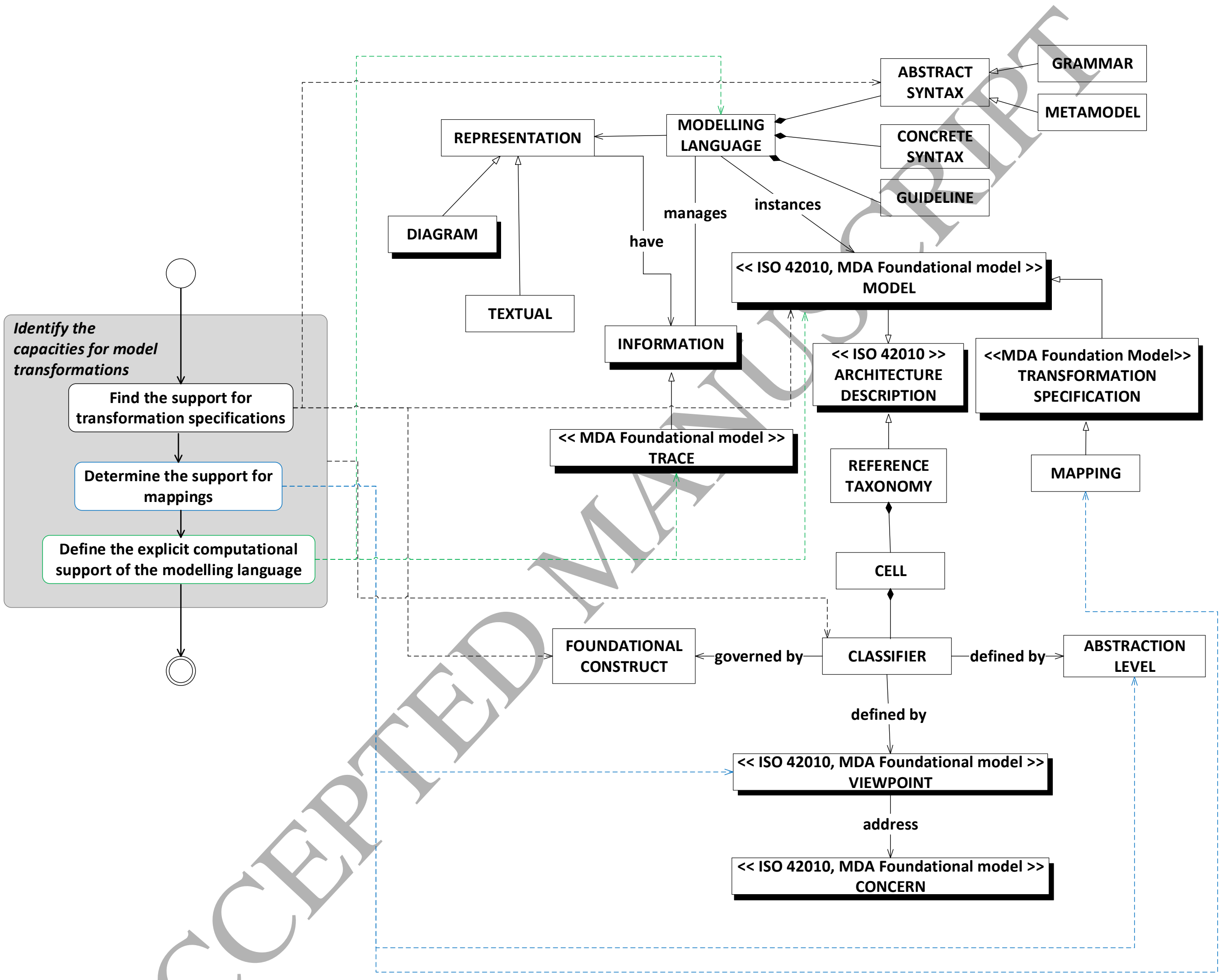

Identify the explicit traces that support the navigation between abstraction levels / viewpoints
Find the traceability mechanism
Identify the native support for each abstraction level supported
Identify explicit mechanisms for taxonomic metamodel preservation
Reason about the justification of multi-viewpoint supported relationships
REPRESENTATION
MODELLING LANGUAGE
ABSTRACT SYNTAX
GRAMMAR
METAMODEL
CONCRETE SYNTAX
GUIDELINE
DIAGRAM
TEXTUAL
have
manages
instances
<< ISO 42010, MDA Foundational model >> MODEL
INFORMATION
<< ISO 42010 >> ARCHITECTURE DESCRIPTION
<<MDA Foundation Model>> TRANSFORMATION SPECIFICATION
<< MDA Foundational model >> TRACE
REFERENCE TAXONOMY
MAPPING
CELL
FOUNDATIONAL CONSTRUCT
governed by
CLASSIFIER
defined by
ABSTRACTION LEVEL
defined by
<< ISO 42010, MDA Foundational model >> VIEWPOINT
address
<< ISO 42010, MDA Foundational model >> CONCERN

Figure A.13: Association of activities of the *Identify the explicit traces that support the navigation between abstraction levels / viewpoints* block with concepts of the MMQEF metamodel.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT



Identify the capacities for model transformations
Find the support for transformation specifications
Determine the support for mappings
Define the explicit computational support of the modelling language
ABSTRACT SYNTAX
GRAMMAR
METAMODEL
MODELLING LANGUAGE
REPRESENTATION
CONCRETE SYNTAX
GUIDELINE
DIAGRAM
TEXTUAL
manages
instances
have
<< ISO 42010, MDA Foundational model >> MODEL
INFORMATION
<< ISO 42010 >> ARCHITECTURE DESCRIPTION
<<MDA Foundation Model>> TRANSFORMATION SPECIFICATION
<< MDA Foundational model >> TRACE
REFERENCE TAXONOMY
MAPPING
CELL
FOUNDATIONAL CONSTRUCT
governed by
CLASSIFIER
defined by
ABSTRACTION LEVEL
defined by
<< ISO 42010, MDA Foundational model >> VIEWPOINT
address
<< ISO 42010, MDA Foundational model >> CONCERN

Figure A.14: Association of activities of the *Identify the capacities for model transformations* block with concepts of the MMQEF metamodel.

ACCEPTED MANUSCRIPT

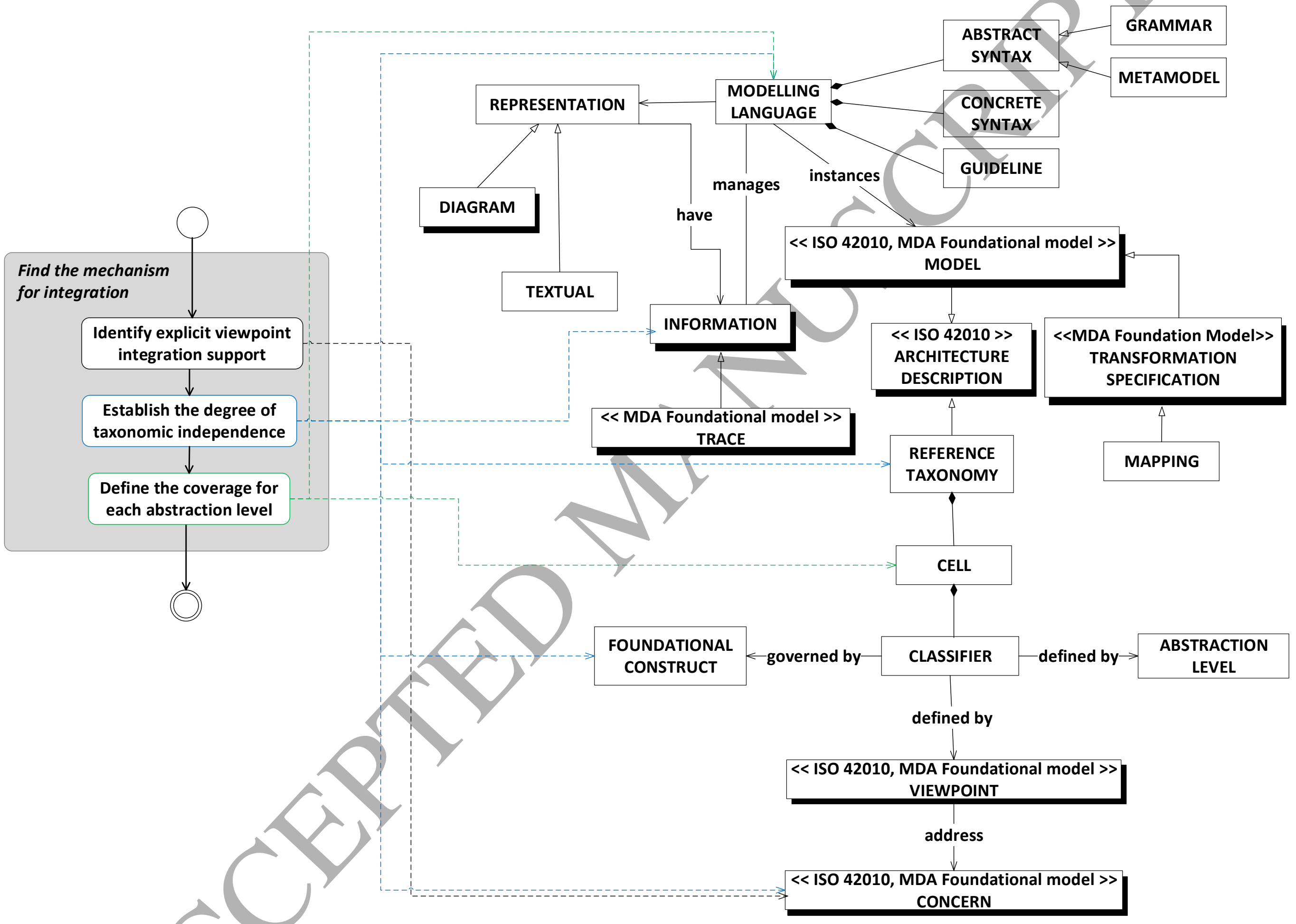
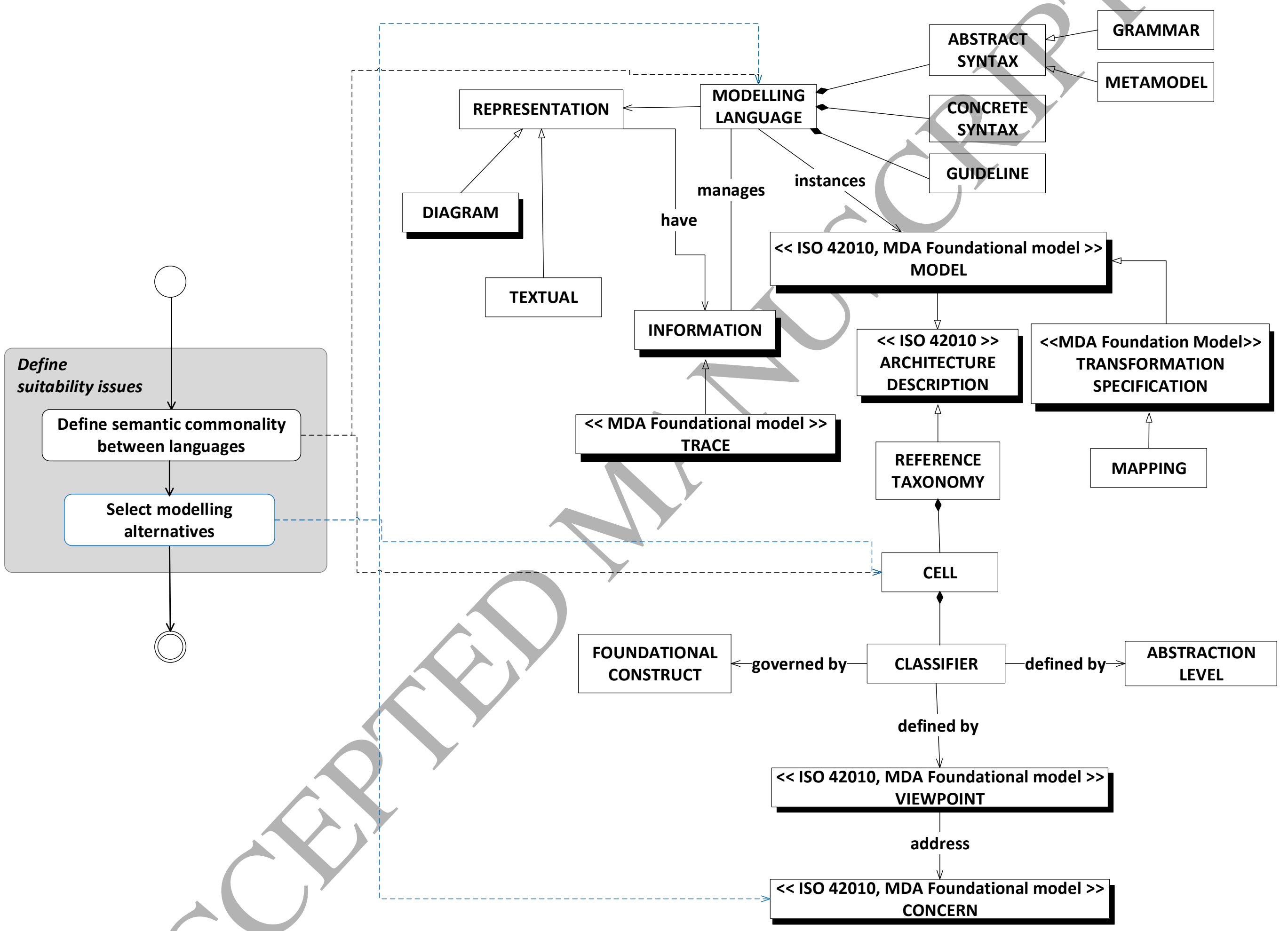

Find the mechanism for integration
Identify explicit viewpoint integration support
Establish the degree of taxonomic independence
Define the coverage for each abstraction level
REPRESENTATION
MODELLING LANGUAGE
ABSTRACT SYNTAX
GRAMMAR
METAMODEL
CONCRETE SYNTAX
GUIDELINE
DIAGRAM
TEXTUAL
have
manages
instances
<< ISO 42010, MDA Foundational model >> MODEL
INFORMATION
<< ISO 42010 >> ARCHITECTURE DESCRIPTION
<<MDA Foundation Model>> TRANSFORMATION SPECIFICATION
<< MDA Foundational model >> TRACE
REFERENCE TAXONOMY
MAPPING
CELL
FOUNDATIONAL CONSTRUCT
governed by
CLASSIFIER
defined by
ABSTRACTION LEVEL
defined by
<< ISO 42010, MDA Foundational model >> VIEWPOINT
address
<< ISO 42010, MDA Foundational model >> CONCERN

Figure A.15: Association of activities of the *Find the mechanism for integration* block with concepts of the MMQEF metamodel.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT



Define suitability issues
Define semantic commonality between languages
Select modelling alternatives
MODELLING LANGUAGE
REPRESENTATION
DIAGRAM
TEXTUAL
ABSTRACT SYNTAX
GRAMMAR
METAMODEL
CONCRETE SYNTAX
GUIDELINE
manages
instances
have
<< ISO 42010, MDA Foundational model >> MODEL
INFORMATION
<< ISO 42010 >> ARCHITECTURE DESCRIPTION
<<MDA Foundation Model>> TRANSFORMATION SPECIFICATION
<< MDA Foundational model >> TRACE
REFERENCE TAXONOMY
MAPPING
CELL
FOUNDATIONAL CONSTRUCT
governed by
CLASSIFIER
defined by
ABSTRACTION LEVEL
defined by
<< ISO 42010, MDA Foundational model >> VIEWPOINT
address
<< ISO 42010, MDA Foundational model >> CONCERN

Figure A.16: Association of activities of the *Define suitability issues* block with concepts of the MMQEF metamodel.

ACCEPTED MANUSCRIPT